TURING

自然计算

DNA、量子比特和智能机器的未来

[美] 丹尼斯·萨莎　凯西·拉瑟　著

金从军　仇祝平　译

Natural Computing

DNA, Quantum Bits, and the Future of Smart Machines



人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

作 者 简 介

丹尼斯·萨莎（Dennis Shasha）

纽约大学科朗研究院计算机科学教授。先后获得耶鲁大学理学学士学位、雪城大学理学硕士学位和哈佛大学理学博士学位。目前正与生物学家合作，研究微阵列分析、组合设计和网络推理方面的模式发现。

凯茜·拉瑟（Cathy Lazere）

获得耶鲁大学文学学士学位，纽约大学工商管理硕士学位。自由撰稿人，曾任经济学人智库编辑。她为大型金融服务机构、咨询公司及律师事务所撰写了大量有关公司财务及技术方面的文章。

译 者 简 介



金从军

本科专业为物理学，做过大学教师、创业者、渠道总监、程序员、开发项目经理、IT培训教师等。曾编写教材，翻译过计算机语言相关的图书及技术资料。一直在寻找自己喜欢做的和擅长做的事情，游戏与编程，酿酒与咖啡，中医与瑜伽，这些是我的最爱。



仇祝平

曾就读于安徽医科大学和中国科学技术大学，生物物理学博士。爱阅读，也希望能通过自己的努力，为读者提供好书。

TURING

自然计算

DNA、量子比特和智能机器的未来

[美] 丹尼斯·萨莎　凯茜·拉瑟 著
金从军　仇祝平 译

Natural Computing

DNA, Quantum Bits, and the Future of Smart Machines

人 民 邮 电 出 版 社
北　京

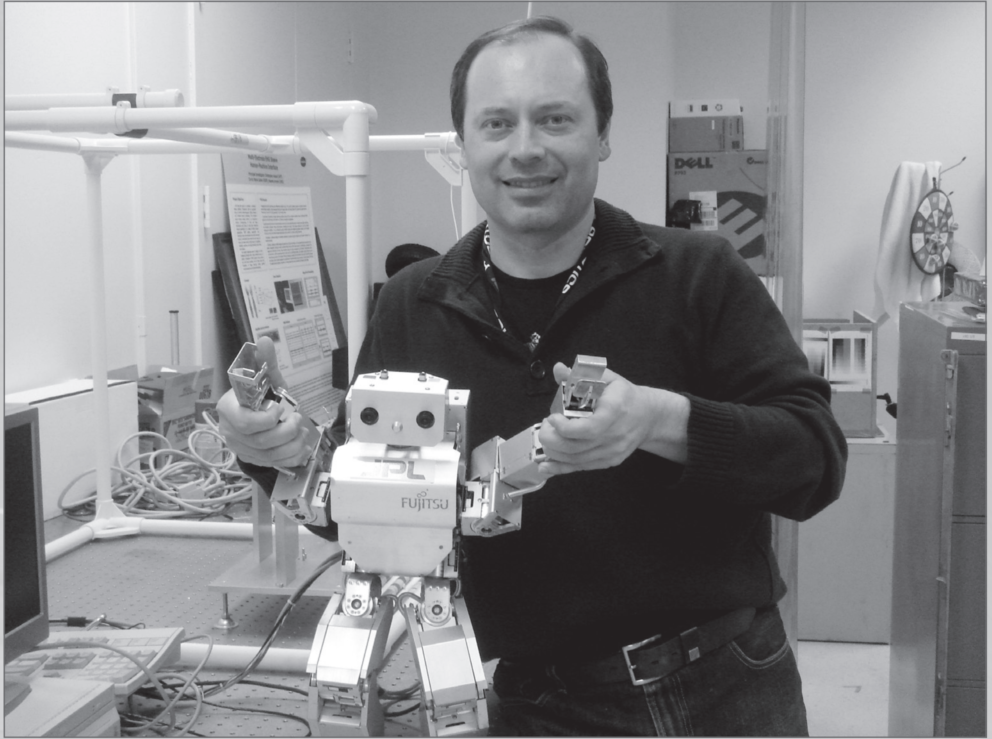


罗德尼·布鲁克斯（第 1 章）站在马文·明斯基和他本人的合影前。（凯茜·拉瑟 摄）



斯托伊卡（第 2 章）小时候在摩尔达维亚。

（阿德里安·斯托伊卡 提供）



阿德里安·斯托伊卡和一位特殊的朋友。（阿德里安·斯托伊卡 提供）



微笑的阿德里安·斯托伊卡手拿一个尚未学会微笑的进化电路。



路易斯·奎尔斯（第 3 章）站在橡树岭国家实验室的一个碳原子核里。（凯茜·拉瑟 摄）



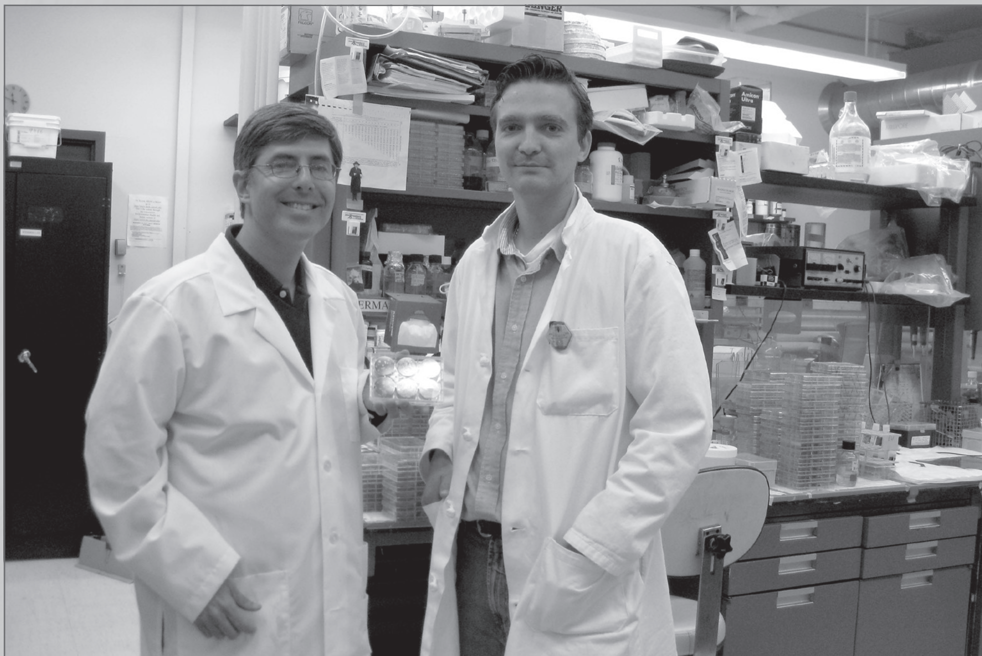
冲浪高手洛夫莱斯和职业网球手巴拉姆比（第 4 章），巴拉姆比手里拿着一种与众不同的记分牌。（凯茜·拉瑟 摄）

内德·希曼（第 6 章）
在纽约大学的办公桌旁。
（内德·希曼 提供）





保罗·罗特蒙德（第 7 章）坐在加州理工学院的 DNA 泳池前。（凯茜·拉瑟 摄）



史蒂夫·斯凯纳（第 8 章）与分子遗传学和微生物学系的同事斯蒂芬·米勒，他们自豪地展示合成脊髓灰质炎病毒的噬菌体。注：科学家们穿着白大褂，但不提供穿着防护服的照片。

（凯茜·拉瑟 摄）



史蒂夫·斯凯纳站在印度众神旁边。（凯茜·拉瑟 摄）



盖瑞·苏斯曼（第 9 章），笔夹子上印着“书呆子骄傲”。（凯茜·拉瑟 摄）

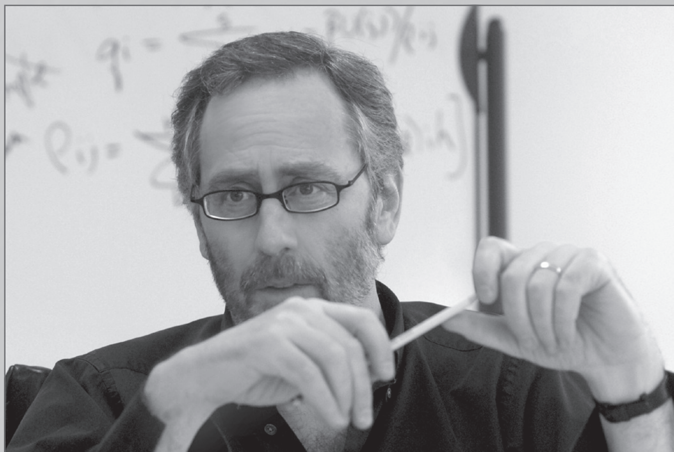


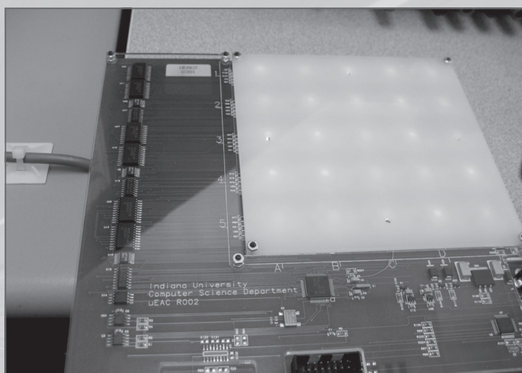
哈佛的桌子都是智能的。拉迪卡·纳格帕（第 10 章）正坐在其中一张桌子旁边。（伊莱扎·格林内尔 摄）



哈佛大学工程和应用科学学院蒙蒂·代诺（第 11 章）证明可以攀上阿卡迪亚国家公园的顶峰。（蒙蒂·代诺 提供）

大卫·肖（第 12 章）和图形计算器。
（大卫·肖 提供）





模拟计算机的主板。（凯茜·拉瑟 摄）



乔纳森·米尔斯和布莱斯·汉博（第 13 章）在布卢明顿的印第安纳大学实验室。
（凯茜·拉瑟 摄）



斯科特·阿伦森（第 14 章）和他的“复杂性动物园”。（凯茜·拉瑟 摄）



斯考特·阿伦森最喜欢的证书：他的“逃离高中得自由”卡。（凯茜·拉瑟 摄）

图书在版编目（CIP）数据

自然计算 ：DNA、量子比特和智能机器的未来 /
（美）萨莎（Shasha,D.），（美）拉瑟（Lazere,C.）著 ；
金从军，仇祝平译. -- 北京 ：人民邮电出版社，2014.9
ISBN 978-7-115-36311-4

Ⅰ. ①自… Ⅱ. ①萨… ②拉… ③金… ④仇… Ⅲ.
①人工智能－计算－文集 Ⅳ. ①TP183-53

中国版本图书馆CIP数据核字(2014)第182045号

内 容 提 要

本书介绍了16位致力于解决计算领域前沿问题的科学家，他们分别在科学、工程金融等领域从事极富挑战性的工作。本书记录了与这些科学家的对话内容，描绘了新的计算机架构和丰富多彩的新型软件技术。书中每章自成一体，揭示了这些科学家独特的探索之路；同时，还介绍了作者写作过程中产生的一系列奇思妙想，这些思想注定会让这个世界变得更好。

本书适合所有对未来智能机器及未来计算感兴趣的读者阅读。

◆ 著 [美] 丹尼斯 · 萨莎 凯茜 · 拉瑟
译 金从军 仇祝平
责任编辑 李松峰
执行编辑 李 静 李 瑛
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京 印刷

◆ 开本：720×960 1/16
印张：12.5 插页：8
字数：168千字 2014年9月第1版
印数：1－4 000册 2014年9月北京第1次印刷
著作权合同登记号 图字：01-2013-3658号

定价：45.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

版 权 声 明

本书作者的其他著作

丹尼斯·萨莎和凯茜·拉瑟还著有：

《奇思妙想：15位计算机天才及其重大发现》

丹尼斯·萨莎还著有：

《程序员面试逻辑题解析》

The Puzzler's Elusion: A Tale of Fraud, Pursuit, and the Art of Logic

Puzzling Adventures: Tales of Strategy, Logic, and Mathematical Skill

Doctor Ecco, Mathematical Detective (Originally Titled Codes, Puzzles, and Conspiracy)

Doctor Ecco's Cyberpuzzles: 36 Puzzles for Hackers and Other Mathematical Detectives

The Puzzling Adventures of Doctor Ecco

献给在引领科学进步过程中不断制造问题的人。

前　　言

在筹划这本书的时候，我们准备与一些科学家进行对话，他们一直致力于解决计算领域的前沿问题。我们希望他们能描绘出新的计算机架构和丰富多彩的新型软件技术。书中介绍的这16位科学家，有的可以控制万里之遥的宇宙飞船，有的能给细菌植入智能，还有的在搭建快速计算机，运行速度可以媲美百万台式机的组合。这些科学家分别在科学、工程甚至是金融领域从事极富挑战性的工作。我们对这多样化的背景早有准备，但有一点没有想到的是，所有这些领域最终都浮现出一幅共同的图景：与自然相融合，是计算领域未来的发展方向。

这里有三条主要的线索可以支持上述观点。

首先，生物学思想为数字计算提供了新思路。尽管它尚未影响到常用的文字处理软件或数据中心，但在一些需要将技术发挥到极致的应用中，却已经得到体现。例如，用计算机控制宇宙飞船的整个飞行及着陆过程。飞船一旦起飞，就难以再对其部件施以人工维修，但是富于创新的飞船工程师提出，可以设计一种能自我修复的机器。如果不是身处工程领域，你恐怕无法领会这个思路变化的深意。不是试图造一台高精度的机器去应付所有可能的问题，而是构造一台能够自适应的机器，它甚至能应付设计者们都想不到的一些问题。

对于如何实现这一新的设计理念，还存在一些争论，但本书介绍的这些科学家们认为，这一理念中应该包含进化或某种形式的学习过程。乍一看，进化与学习似乎非常不同：进化体现在多代生物体中，而学习和适应只发生在单一生物体中。但是从道理上讲，二者是非常相像的：先尝试，然后观察反馈结果，再根据反馈继续新的、可能更好的尝试。我们会把学习与有意识的改进行为联系起来，而进化的产生则是下意识的，是一种化学反应，甚至要经过新陈代谢。但是就生存这一目的而言，学习和进化具有同样的效果。

许多应用都基于进化技术。在本书中，一位冲浪高手竟然变身为金融数学大师，用进化算法计算出何时可以买进或卖出美国国债；一位教授通过检查安全程序的容错性，分析出导弹防御系统的安全性。在这两个例子中，进化、学习和适应都是相互关联的。

其次，生物体可能会取代硅片。基于DNA或细菌细胞的计算几乎是零成本的（数十亿细菌仅靠一丁点儿糖水就可以生长），而且DNA计算具有巨大的并行能力。终有一天，活体细菌而不是那些硅电子，将在我们的身体或微型工厂的内部进行运算。这种利用生物进行计算的方法，需要发展出一种全新的计算范式。人们不再需要花几年的时间来手工设计那些性能稳定的计算机，也不再需要给它们编制好统一的名称，安装在气势恢宏的空调机房里。

由数百万的细菌或病毒组成的计算机，没有名字，而且分散在各地，它们只与自己的邻居交流，还会经常出错。让它们来做有益的事，不仅让人担忧，而且似乎绝不可能。但这的的确确是可能的，我们有数学家们所说的“存在性证明”：人类的身体由100万亿个细胞组成，人能跑，能思考，还能恋爱。而这些事情，单个细胞却做不到。

再次，新的应用类型促使我们反思计算的基本机制。模拟蛋白质折叠，或破

译密码，其运算量之大，需要几千个处理器协同工作。与晶体管的速度相比，计算机内存和交换机间的通信则要慢很多，这种协同工作策略，只适合于相隔较远的处理器间极少通信的情形。因此，这个星球上最快的数字计算机，在设计之初，就不是用来传递信息的。

举个例子，有一位拥有千万资产的发明家，他开始着手设计一台模拟蛋白质行为的机器。为此，他用硬件把蛋白质中每个原子的相关信息，直接通过电路移动到特定的位置。在这些位置上，这些信息与相邻原子的对应信息相结合：所有信息不必返回到计算机的低速内存中。这样做的结果可以使计算速度提升一千倍，足以使一项原本需要一个世纪才能完成的计算任务，在不足一百天内完成。

另一位设计者资金没那么充足，他搭建了一台计算设备，主要计算单元由一片连接了25根导线的泡沫组成。通过测量电流的方法，可以模拟微分方程，来表征大量的有关“连续性”的科学问题，从银河系的恒星轨道预测，到蝴蝶翅膀着色的衍变。这台“扩展的模拟计算机”彻底颠覆了计算的概念。数字计算机是使用0、1及算法来“计算”答案，而它是在“测量”答案。

你可能发现自己开始喜欢这些机器了。与那些用于计算、发邮件或玩象棋的坚硬的金属装备比起来，这样的机器更加人性化：它们能自我修复，挑战超难问题，有时也会犯错。有一天，或许这样的计算设备可以帮你接续断骨、保持桥梁的稳定，甚至帮助你在水下呼吸。

在本书的写作过程中，我们触碰到了一系列的奇思妙想，这些思想注定会让这个世界变得更好。书中的每一章都描绘了一段独特的探索之路。希望你喜欢这些爱冒险的探险者的故事，就像我们享受写书过程一样。

致　　谢

用文字解释科学往往力有未逮。为了帮我们将复杂的概念转换成通俗易懂的图像，我们请年轻有才华的艾登・戴利画了每章开篇的漫画和既生动又精辟的图表。我们还要感谢两位优秀读者的意见：亚伦・阿普尔和格伦・巴特弗斯。

家人和很多朋友都一直鼓励着我们，包括克劳德・谢罗、汤姆・肯特、苏菲・肯特、安・梅特卡夫、卡拉・佩格莱尼和理查德・所罗门。

诺顿出版社（W.W.Norton）是每一位作者的梦想。出版过程中的每个阶段都得到了力求完美的专业人士亲切、圆满的处理。我们要感谢编辑布兰登・卡利的好品味和坚毅；挑剔而沉着的文字编辑斯蒂芬妮・希伯特耐心地比对我们的改动和她自己的改动。此外还有制作人戴文・扎恩跟设计师布莱恩・穆里根，很荣幸和他们一起工作。

最后，还要向本书中描写的所有科学家致以特别的谢意。他们非常慷慨地贡献了自己的时间，跟我们分享了他们的精彩想法以及对未来的展望。

目　　录

第一部分　自适应计算

第 1 章　罗德尼·布鲁克斯 …… 7
动物法则

第 2 章　格伦·里夫斯和阿德里安·斯托伊卡 …… 17
为遥远的行星而设计

第 3 章　路易斯·奎尔斯 …… 33
为设计团队注入进化思想

第 4 章　杰克·洛夫莱斯和阿穆特·巴拉姆比 …… 43
追赶大潮

第 5 章　南希·莱韦森 …… 55
“这是系统，笨蛋”

第二部分　驾驭生命物质

第 6 章　内德·西曼 …… 73
生命的边际

第 7 章 保罗·罗特蒙德 ······ 83
用生命物质模仿艺术

第 8 章 史蒂夫·斯凯纳 ······ 95
编程漏洞

第 9 章 杰拉尔德·苏斯曼 ······ 105
建造十亿台生物计算机

第 10 章 拉蒂卡·纳格帕 ······ 115
从局部到整体

第三部分 物理和速度

第 11 章 蒙蒂·代诺 ······ 127
速度缔造者

第 12 章 大卫·肖 ······ 137
安东和巨大的飞秒镜

第 13 章 乔纳森·米尔斯 ······ 147
自然而然

第 14 章 斯科特·阿伦森 ······ 161
寻找物理新法则

后记 ······ 174

自然计算时间线 ······ 177

人名索引 ······ 180

参考文献 ······ 183

第一部分

自适应计算

现代工业始于“零件可互换”概念，最早可追溯到15世纪约翰内斯·谷登堡的活字印刷术。到了18世纪，制造商们越发关注零件的精度。伊莱·惠特尼的可互换步枪部件的精度是1/30英寸（大约1毫米），而现行的机械公差通常为10微米（百万分之一米），百倍于惠特尼能够实现的精度。光学公差测量更是达到了纳米量级，百万倍于惠特尼的精度。现在，设计师们有条件制造超高精度的机器，来满足任务的需要。

主流的计算机科学以算法为基础。所谓算法，指的是这样一种方法，它确保计算机按照指定的效率，针对大量请求给出正确的回应。可以把算法想象成食谱：按照指定的顺序将某些指定的食材混合在一起，并获得预期的结果，做出一款菜肴。例如，“归并排序”算法，就是按顺序放置所有元素，而不考虑这些元素的类型。

尽管算法始终是计算的核心，但有些问题却根本就没有算法。考虑遇到下面的问题：要在南极生存，你的装备要在零下60℃的状态下也能正常工作。你知道你的住所和衣服可能会遭受任何一种灾难情形，那么你本人还有你的装备该怎样才能生存呢？搞算法的计算机科学家会抱怨说，这个问题提得不恰当。如果灾难过于严重，就无法生存。但是，假如必须设计出一套方案，在绝大多数情况下有

效，你会怎么做呢？你需要内置自适应机制，或者相类似的进化方法。

早在1954年，就职于普林斯顿高级研究院的数学家尼尔斯·奥尔·巴里切利，就用计算机建立了进化的简单模型。大约15年后，德国计算机科学家英戈·雷兴贝格和汉斯–保罗·施韦费尔使用进化改进了复杂工程问题的设计。

在自然界中，进化适用于生物。而在计算世界里，进化适用于设计。两种情况下，进化都能取得出色的成果，而且无需设计师的刻意参与。1975年，密歇根大学的约翰·霍兰德完成了一部具有里程碑意义的著作——《自然与人工系统中的适应》，书中揭示了不同的进化设计方法的共同特征，并使用统一的数学框架对其进行了改进。

霍兰德的工作为现代遗传算法（有时也称为“进化算法”）建立了基础框架，它需要循环使用以下步骤：

(1) 选择一个起始群体，其中包含了各种可能的候选设计（即候选个体）；
(2) 评估每个候选个体的“适应度”得分，依据的可以是个体的金钱支出或者能量消耗；
(3) 记住“适应度”得分最高的候选设计；
(4) 选取具有最佳“适应度”的候选设计，通过以下两种方法创建一个新的群体：对它们做随机的微调；通过与其他候选设计重组来对设计做较大改动。

设想你用这个方法设计一辆汽车。假设一种优选的设计采用复合底盘并搭载六缸发动机，而另一种优选的设计则采用铝合金底盘及电动发动机，那么组合起来的设计可能是采用复合底盘加电动发动机。亲代设计相结合所产生的子代方案，综合了亲代的某些特征。

尽管进化可以带来更优的设计，但小范围的适应更加省事。例如，没有进化，

你也可以学会骑单车或杂耍。这种水平上的适应会伴随着尝试和失败，但生物体自身却不需要变化。罗德尼·布鲁克斯研究运动中的适应问题。自上世纪80年代以来，他利用自适应机制，设计出能够智能移动的机器人。他从昆虫、大象和壁虎那里获得灵感，在开创性的研究过程中，给机器人的“智能”做了重新定义。

设想你正在为一个机器人设计软件，让它能在另一个行星的表面行走。你无法准确地知道它的处境。你知道行星的地表凹凸不平，而且环境极端恶劣，但并不知道具体细节。事实上，你面临的不仅是“未知”，而且还是NASA喷气推进实验室的格伦·里夫斯所说的“对未知的无知”：你甚至无法描述这些“未知”。凭空地设计并发射一个探测车，然后祈祷一切顺利，这是不可行的。事实上，目前这项技术的要求是要跨越一亿六千公里进行远程诊断。为此，里夫斯必须为航天器设计一个仪器，这个仪器就像一个唠唠叨叨的病人，不停地汇报自己的近况，感觉不错时还向你发送报告。当仪器“生病”时，他的团队要用“补丁”（即对软件做些小修改）的方式发送“电子假肢”来修复它。设备用“补丁”来替代错误代码，就像病人换上了假肢一样。

阿德里安·斯托伊卡也在喷气推进实验室工作，他设想未来的航天器具有自愈能力。试想这样的极端考验：以火星为例，地表的昼夜温差可以从极寒的零下133℃到舒适的27℃。人可以随温度变化而增减衣服，电路却不能，但或许它可以改变电流的流动方式。斯托伊卡设计的电路能够自行“进化”出解决方案。他梦想有一台设备，能够靠进化适应存活百年。

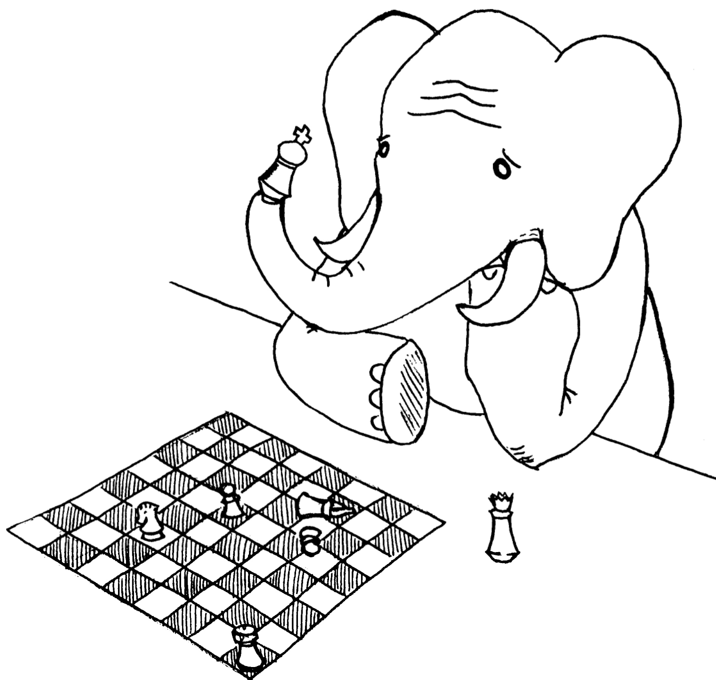
有些人对于遗传算法享有“算法”的美誉表示怀疑，因为遗传算法不能保证计算的效率和正确性。这是事实，遗传算法不提供任何保证。但对于那些没有已知算法的问题，它却常常能给出令人惊奇的优化设计。利用遗传算法，路易斯·奎尔斯设计了可定制的、能适应极端环境（包括太空）的核电站。为了获得核电站的设计规格，如输出功率、重量（如果为航天器提供动力就必须考虑）等，他找

专家讨论，并从他们那里得到设计参数，然后尝试给出一个设计。设计必须在满足规格要求的前提下，兼顾各项参数之间的平衡。为了做到这一点，他需要在数万亿个可能的设计中进行选择。手工方式只能在少量的设计中进行筛选。而且他意识到自己经常会走回头路，因为他会比较眷顾那些研究过的设计方案。相比之下，当他用计算机运行遗传算法程序时，却经常可以获得出人意料的设计，无论是制造成本，还是性能，都要好于他以往的设计。还有，如果设计规格发生变化（这会经常发生），他只要重新启动程序就能获得新的设计，不会产生人为偏向。未来很有可能正是要用这种方式来设计极端复杂的工程制品。

杰克·洛夫莱斯和阿姆鲁特·巴哈拉勃将类似的思路应用于金融领域，用遗传算法来设计规则，帮助他们进行国债交易。当工程师们用复杂的物理指标来评价“适应度”时，洛夫莱斯和巴哈拉勃则使用最基本的金融评判标准：以较低的风险获取较高的收益。他们用历史数据来评估一个规则的优劣，如此庞大的搜索空间（所有可能的规则数）使他们无法理解遗传算法生成的这些规则，但这个方法很有效。

南希·莱韦森更多地在研究人性中与尖端技术相互纠缠的自然因素。她研究诸多的工程杰作——发电厂、导弹防御及航天器，努力使它们更加安全。她最早是从事计算机研究的，但不久之后便动身前往新几内亚丛林，做她非常感兴趣的认知心理学研究，回来之后，对于计算机有了全新的视角。她相信安全必须以可靠的适应作保证：当错误发生时，系统必须能够对错误进行补救。按照莱韦森的观点，所谓“系统”，不能仅限于软件说明书中所规定的边界，而必须扩展到与人类管理相关的所有链路中。当计算要素与人身安全相关时，适应性就是多级的反馈与修正。一次低级的故障，必须得到人工或机器的高级补救。莱韦森的方法试图确保系统的每个级别都能侦测到下层问题，并做出适当的响应。不难理解，这个方法可以帮助我们预防意外的导弹发射及空中交通事故。





■ ■ ■ ■

人们通常会假设机器人内置了一系列的逻辑表达，但是假如根本不存在这样的逻辑表达，情况又会怎样呢？我曾经观察过昆虫的行为……它们真能把周围的世界渲染成三维吗？或者说，在它们仅有五万个神经元的小脑袋里，真的有一个计算机绘图模型吗？这是那些神经元的作用吗？

——罗德尼·布鲁克斯

| 第1章 |

罗德尼 · 布鲁克斯

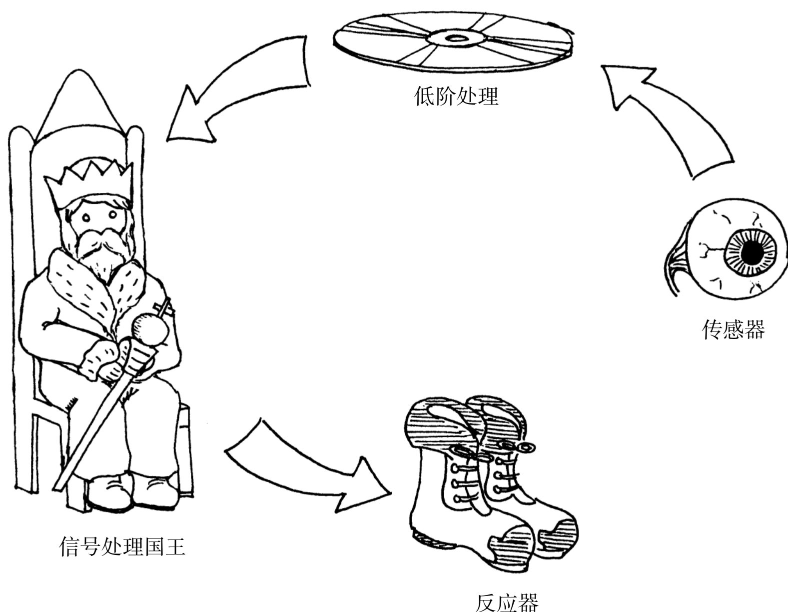
动物法则

自人工智能（AI）于20世纪50年代诞生以来，其在智商测试及下棋中的卓越表现，似乎成了“智能”的代名词，不过这只是学术界的评价。此后，智能又被附加了许多其他的定义，包括情商和霍华德 · 加德纳所提出的人际智能与运动智能等测量指标。但是，如果把智能定义为在这个世界上的生存能力，我们就需要考察更多的基本技能。我们能走路、识别物体、绕过障碍物，这是怎样做到的呢？你会说：“凡是动物都做得到！”罗德尼 · 布鲁克斯会同意说：“对极了！”事实上，如果机器人不必完全效仿人类的行为，那么它们会表现得更出色。举个例子，谁更适应在崎岖的地面上行走，是人类还是昆虫？如果你见过昆虫从难以想象的小洞中爬出，你会把票投给昆虫的。

在1990年发表的一篇影响深远的论文“Elephants Don’t Play Chess”（大象不玩象棋）中，罗德尼 · 布鲁克斯从进化的角度表达了这样的观点：人类的所谓高智商相对而言其实是微不足道的。生命在地球上的出现距今35亿年，他强调，脊椎动物和昆虫出现在最近的10%的时间里，大约4.5亿年前。类人猿直到最近的0.5%的时间，即1800万年前才出现。而农业仅有19 000年的历史，是地球生命历史的0.000 5%。专业学科的出现只是近几百年的事情。

计算机卓有成效地展现了人类在过去几百年的历史中所掌握的技能，或许是因为我们最在意自身的这些技能，并且为之付出了极其自觉的努力。但是我们的某些本能行为却给计算带来了巨大的挑战。太空旅行不过只有很短暂的历史，但是写一段引导飞船前往火星的程序，远比造一个像山羊一样能跨越崎岖道路的机器人，要来得容易。山羊的技能经历了数十亿年的进化，但人类智能的进化只用了几百万年。这样相差千倍的时间尺度让我们在这些“原始”的智能面前自惭形秽。

在布鲁克斯写他的“大象”论文时，人工智能领域惯于把智能模拟为符号处理。科学的目标是设计出像视觉系统这样的传感器模块。它们将获取的外部世界信息转变成符号，并将符号传递给一个智能核心：一种电子国王。这个国王可以处理这些符号，并指挥执行器（通常是轮子）移动。在很多方面，这个渐进的过程借鉴了大企业或军队中的理想化层级结构：“首脑”在上，眼睛和四肢在下。无论是从哲学的角度考虑，还是出于实用的目的，布鲁克斯都反对这种范式。



国王模型：国王收到传感器数据，推算数据，并发出决定给反应器付诸实施

布鲁克斯从哲学角度反对这一范式，可能源于他进入科学研究和MIT非同寻常的经历，现在，他已经是MIT计算机科学与人工智能实验室（CSAIL）的教授。布鲁克斯1954年生于澳大利亚的阿德莱德，那里虽然不算偏远的内陆，但距离计算机科学研究的中心却很远。不过偏远却成为布鲁克斯的优势。没有人告诉他通往专业领域的正确道路。8岁时，他就开始独自设计能玩游戏的计算机；12岁时，他利用旧电话的继电器造了一台能玩一字棋游戏的简易计算机。他曾立志要从事游戏设计工作。

1972年，布鲁克斯开始到南澳大利亚的弗林德斯大学学习。周末他获准使用大学里唯一的一台计算机，这台机器只有16 K字节的内存以及1 M字节的磁盘，而1 M字节只相当于现在台式机内存容量的千分之一[①]。尽管如此，1 M字节要远大于16 K。布鲁克斯设法为计算机编写程序，在必要时可以清除磁盘上的数据，用磁盘空间来扩充内存，这样，他相当于有了1 M字节的内存。他使用了一种叫做“虚拟内存”的新发明，前几年才商用化的一种技术。布鲁克斯没有去查找与这项技术有关的论文，但他自己做出来了。“有人跟我提起过这个想法，”他说，“听起来很不错，于是我在这台电脑上实现了虚拟内存。”

在斯坦福读博士期间，布鲁克斯结识了另一位勤奋进取的机器人专家——汉斯・莫拉韦克，一个比他高几届的研究生。1979年夏天，在校园后山上的一间小实验室里，布鲁克斯帮助莫拉韦克完成了一个称作Cart的机器人。每个深夜，当其他人都回到了家里，只有布鲁克斯和莫拉韦克还在安装Cart。机器人在深夜的6个小时里，可以绕着实验室移动大约20米。这两个年轻的科学家想要给机器人创造立体视觉——通过在两个观察点上提供略有不同的图像，使机器人形成关于深度的知觉。立体视觉通常需要两台摄像机，但在那个年代，摄像机很昂贵，他们

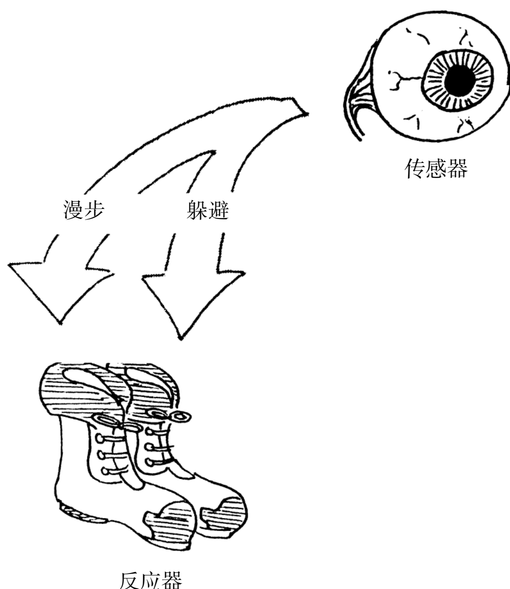
① 原书为“百万分之一”，现在主流内存1 GB=1024 MB。——译者注

只有一台。于是他们只能在两个观察点之间来回切换摄像机。布鲁克斯说："我就像一个搬运家具的勤杂工一样，把设备拆下来，再连上。"这台计算机先看一眼这个世界，接着计算15分钟，然后移动1米。再睁开眼睛，看一下，合上眼睛算15分钟，然后朝着它认定的目标，闭着眼睛再行走1米。布鲁克斯解释说："那时，我认为这已经是超大量的计算了。"

布鲁克斯的博士论文是关于机器视觉的，那是一种传统的人工智能方法，摄像机负责向计算机传递图像。布鲁克斯的程序将视觉场景转换为符号，再由一个假想的"智能"——符号处理国王进行处理。他说："毫无疑问，要先有一台摄像机，然后从摄像机中获得像素，最后把像素转变为现实世界的逻辑描述，这就是基本思路。"

1988年，布鲁克斯到泰国度假，来到他第一任妻子的老家，老屋是一座位于河边的吊脚楼。家里人不会说英语，于是布鲁克斯一个人坐在那里观察昆虫。他边看边琢磨，心中开始怀疑这种符号化的人工智能范式。他就是不能相信，在昆虫那个"仅有5万个神经元的小脑袋里"，能够形成逻辑描述。

1990年，布鲁克斯的"大象"论文解释了他戏称的"新式AI"。他的前提是：对一个智能系统的表述必须构筑于真实的物质世界之上。用他自己的话说，就是"这个世界是它自己最好的模型"。世界是鲜活的，并包含了所有必要的细节。这意味着布鲁克斯的机器人将摒弃"传统"AI的层级结构，放弃对世界的符号化表征。相反，"新式AI"机器人将掌握一系列彼此独立的预设技能。如同一个人使用同样的四肢和同一双眼睛来玩篮球和走路一样，机器人用共同的传感器和执行器来获得多种技能。有些技能是独立的，尤其是那些最高级的技能，因此即使出现失败，也不会导致其他技能停止工作。



新式AI：机器人首先检测，然后按照简单的规则（如防撞或漫步）移动

为了体现这一理念，MIT的布鲁克斯实验室在1985年制造了一个早期的机器人Allen，这是故意借用了符号化AI的早期支持者Allen Newell的名字。机器人Allen有三个技能：防撞、随机漫步、到达远处目标。布鲁克斯回忆说："Allen很高兴坐在屋子中间，等到有东西靠近它才匆匆跑开，同时避免被撞到，在它的内部，每次返回的声纳信号都代表一个排斥力。"

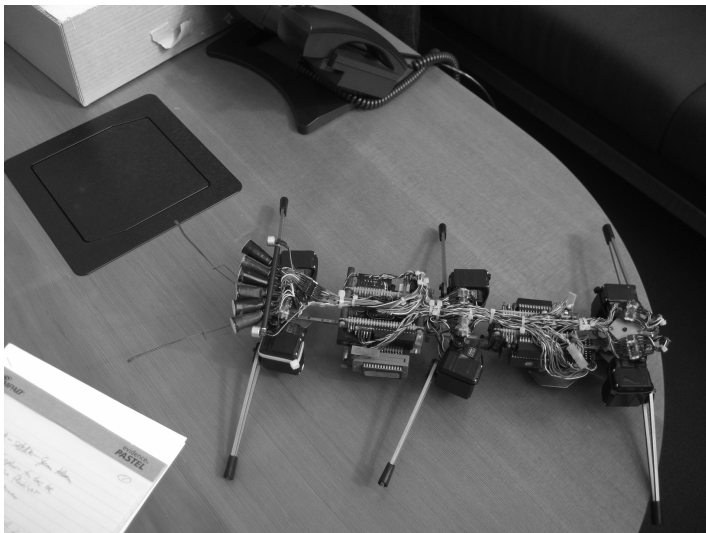
在最低级的状态下，Allen表现得像一只胆小的老鼠，遵循最基本的规则：避免冲撞和被冲撞。如何避免Allen一直躲在角落里呢？原来，每隔10秒钟Allen会收到一次随机漫步指令。注意到漫步也需要用轮子来移动，依照布鲁克斯的策略，不同技能（防撞和漫步）共用同一个执行器（轮子），这里的关键在于防撞技能要优先于漫步技能。

在Allen的第三个功能中，机器人用声纳搜寻远处目标并试图到达那里，同时用里程表来测量行走的距离。像登山运动员一样，在完成比赛的同时要避免从路

边滑落，机器人要将目标搜索技能与基本生存技能结合起来。这项研究看似过于简单，布鲁克斯说："我主张简单。要摆脱那些令人费解的方程。"这一观点与他的某些同事及批评家的信条正好相反，他们坚信越复杂越好。对于布鲁克斯来说，对事情的解释如果必须借助于一堆复杂的数学，那么这个方案将"非常不稳定"。他说："我乐于建造那些不会停工的东西。"

机器人的滚动就算作一种。采用行走方式的机器人是否适合在崎岖的地面上攀援行进呢？在泰国的所思所想即将付诸行动。布鲁克斯与科林·安格尔和格林内尔·穆尔（一名高中生）合作制造了一台有六条腿的行走机器人，取名为Genghis。布鲁克斯回忆说："那时很少有行走机器人，别的会行走的机器人都很大，怕摔。"

布鲁克斯仔细观看了昆虫奔跑的高速视频，"它们一直在摔跟头，反复磕到自己的'下巴'。"他说。由于昆虫的体型很小，力量–体重比很大，因此它们真的可以说是非常糟糕的步行者。相反，肯塔基赛马会上的小母马却经不起摔跤：它可能会因为摔断腿而被迫退赛。



布鲁克斯的行走机器人Genghis，一个有六条腿、不怕摔的低矮设备

在喷气推进实验室（JPL），布鲁克斯为Genghis找到了用武之地。他开始参加他们讨论新的火星探测任务的会议。JPL正在升级一台叫Robbie的机器人，它有一吨重，长着一支巨大的前臂，每分钟只能移动1厘米。JPL估算说要把Robbie送上火星，任务的预算大约是120亿美元。布鲁克斯知道他们不可能拿到这笔经费。开会时，他建议发送一个小型的机器人来代替这个大家伙。“我记得JPL的一位负责设备制造的终身员工说：‘科学家为这个任务等了15年到20年，他不想要一个小不点儿，他想要一台大设备。你不能打发一台小机器人去，你必须发送一台大机器人。’”在谈起这段故事时，布鲁克斯一改他的澳洲口音，用南方口音慢吞吞地说。

布鲁克斯建议采用一个折中的方案：发送100个1千克的机器人来代替这个1吨重的机器人，这样可以减少10倍的质量，还可以分散风险。他说：“如果是发送一台1吨的机器人，你必须小心谨慎地关照到所有的事情，一旦搞砸了，120亿美元的投资就白费了。如果你有100个机器人，损失一个也没什么大不了的。”

布鲁克斯还发现了“机器人军团”计划的另一个优势，就是可以批量生产。制作一个小机器人的100个复制品，要比制造一个1吨重的机器人便宜很多。然而一旦这100个机器人到达火星，需要它们做什么呢？“如果只有一台1吨重的机器人，你会一直控制它；但是如果有100个（小机器人），你就不可能控制它们，它们只能自治。那么好吧——让它们自治吧，从一开始就摆脱你的控制。”

布鲁克斯于1989年向英国星际协会提交论文，公开阐述了他的颠覆性的计划。论文标题是“快速、廉价和失控”（Fast, Cheap, and Out of Control）：1997年埃罗尔 · 莫里斯导演用这个标题拍了一部纪录片，而布鲁克斯在片中扮演一个集园艺师、驯狮员、鼹鼠专家于一身的角色。这篇文章引起了NASA科学家唐娜 · 雪莉的共鸣。1989年雪莉安排科林 · 安格尔（当时还是一名本科生）到喷气推进实验

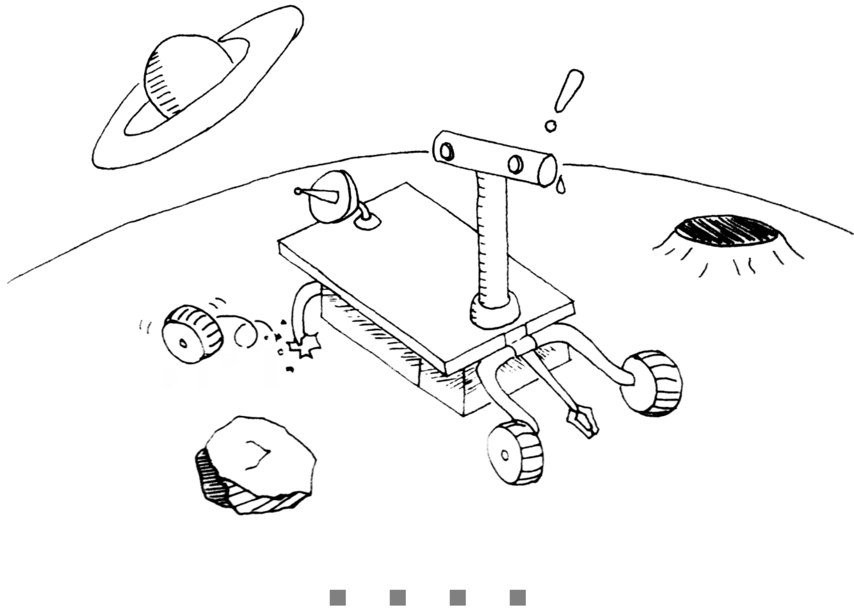
室呆了一个夏天。在那儿，安格尔造了一个叫Tooth的半公斤重的小机器人，Tooth能实现Robbie（JPL的那台1吨重的机器人）的大部分功能。JPL的拉吉夫·德赛及戴夫·米勒使用Tooth的代码造了六个轮子的漫步者Rocky I和Rocky II，现在的火星车就是在此基础上设计而成的。

成功鼓舞了布鲁克斯，他和刚刚毕业的科林·安格尔决定为月球及火星探测任务成立一家公司。他说："我们要把成对儿的机器人送上月球，并贴上我们的广告。"最终，布鲁克斯与阿波罗15号的指挥官大卫·斯科特达成了合作。1992年，布鲁克斯经斯科特介绍，与战略防御计划（SDI）建立了联系，这项计划又称为"星球大战计划"，在克林顿任内被重新命名为"弹道导弹防御组织"。这项计划是政客之间相互博弈的结果。弹道导弹防御系统的开发者们拥有许多被称为"智能卵石"的飞行器，目的是在导弹的发射阶段，利用这些飞行器锁定导弹的运载火箭羽流，并用非核的微型导弹撞击它们。但是到了1992年，苏联的威胁消失了，因此弹道导弹防御组织只好改用其他方式来炫耀其技术了。

新的计划建议将智能卵石放在即将绕月旅行的克莱芒蒂娜号探测器上。布鲁克斯的行走机器人Grendel将替代智能卵石上的导弹探测设备。智能卵石将在月球上着陆，而机器人将承担一些环境测试任务。1993年在爱德华兹空军基地进行了试运行。计划实施在即，政府部门之间却产生了管辖权之争。NASA断言太空探索是它的职责，于是这一计划搁浅了。不过，布鲁克斯的Rocky VI还是被发往火星。NASA似乎与布鲁克斯的观点一致，"他们说，我们要做到更快、更好、成本更低，我们不会再走老路，我们要更加智能。" 布鲁克斯回忆说。然而下一个火星任务却失败了，"更快、更好、更低成本也成了一句空话"。

从那以后，布鲁克斯的iRobot公司自主制造了一款名为Roombas的家用真空吸尘器，同时也为军方生产炸弹拆除机器人。生物工程师罗伯特·富尔与iRobot公

司合作，他们重拾昆虫灵感，用于改进机器人的结构。他们一同研究了蟑螂以及类似的爬行类昆虫的运动，富尔决定使用带有关节的弹簧高跷来模拟昆虫的腿。用这种方式设计的移动机器人可以跨越更加困难的障碍，并且仅用很小的运算量，在没有视觉系统的情况下，也能跳入或跳出坑槽。弹性肢体彻底解决了机器人的移动问题。简约必胜！



■ ■ ■ ■

在地球上使用设备时，无论是诊断还是维修，你都可以实实在在地走近它，触摸到它。然而，一旦设备离开了地球，所有这些都不再可能。它变得遥不可及，你只能通过它传给你的消息来观测它；同样，你也无法去修理它。在这种情况下，如果有东西坏了，你只能要么放弃它，要么采取替代方案。

——格伦·里夫斯

假如要发射一艘历经百年才能抵达目标的宇宙飞船，你绝对不能容忍它携带的电子设备只有不足10年的寿命。你可以尝试携带大量备件，但这有可能让飞船笨重到飞不起来。整个目标的实现完全有赖于这个系统能否依靠自己来延长寿命。

——阿德里安·斯托伊卡

第 2 章

格伦·里夫斯和阿德里安·斯托伊卡

为遥远的行星而设计

想象一下，你必须到客户现场去为一套价值数百万美元的生产系统做故障检测。所有的眼睛都在看着你，客户、你的老板，还有，如果你是主管，那么你的手下也在看着你。为了找到问题所在，你分别检测了软件及硬件的各部分组件，思考再三，尝试着修改数据然后重新运行软件。众目睽睽之下，你得冥思苦想，尝试把现有数据装到其他硬件上去运行，并尽量让自己保持思路清晰、忙而不乱。他们还在看着你，而你又想到了一个或许有效的解决方法，于是还得继续试下去……

现在想象，你的设备在整个应用中仅此一件。此时，看着你的不仅有客户，还有全世界的新闻记者。几分钟前，宇宙飞船已经出发，正在以光速飞离。在最理想的情况下，即天气和电力情况都正常，你可以与故障设备之间进行每秒钟1200比特的通信（DSL有线通信的速度是此速度的5000倍）。你无法更换硬件，你甚至都无法亲眼见到它。

接下来继续想象，这艘宇宙飞船的目的地是一颗遥远的行星，全部行程需要

100年时间。一旦飞船到达那里，就要面临极端的严寒与酷暑。如果零件出了故障，可没法找联邦快递。现在设想一下：这艘飞船能不能进行自我修复？我们欢迎喷气推进实验室（JPL）的格伦·里夫斯与阿德里安·斯托伊卡来接受这个挑战。

修理与学习

格伦·里夫斯生于1960年，在南帕萨迪纳长大。他的父亲是加州交通厅（加州政府部门，负责高速公路相关事务）的一名土木工程经理。当里夫斯还是个小男孩时，就对遥远的太空充满了好奇。在他5岁的生日晚会上，月亮、地球和星星的纸模型挂满了他家后院的树枝头。与此同时，许多更年长的梦想家就住在他家附近——因为帕萨迪纳就是JPL的所在地。

JPL坐落在圣盖博山对面，占地超过两百英亩，由加州理工学院于1930年代创建。在前苏联发射了斯普特尼克号（Sputnik）人造卫星后不久，JPL就设计出了美国的第一颗卫星——探索者一号，并于1958年发射，仅仅比里夫斯出生的时间早两年。JPL实际上是NASA的一个实验室，这里的员工为月球及行星探测计划设计宇宙飞船及用于探测的机器人。

里夫斯回忆说："在我小时候，我们经常开车经过那里（JPL）。那时，妈妈就会指给我看，对我说：'知道吗？有一天你就会去那儿工作。'那时候我并不知道这是个什么地方。"上高中时，里夫斯偏爱自然科学，但不怎么用心。他虽然喜欢物理学，可功课却不够出色。他还参加了一个数学自学小组，但也落在同组人后面。他一心想着成为一名汽车修理工。"我的最大兴趣是找一份工作赚钱，而不是筹划未来的职业发展。"他承认说。

高中毕业时，里夫斯得知他得去读大学——他的父母坚持要这样，但在哪里

读、读什么则由他做主。他本来想去加州大学的圣巴巴拉分校，但这时却爱上了当地的一个姑娘，因此决定去离家更近的波莫纳加州州立理工大学。在那里，他学习了酒店管理、会计学、信息系统等课程。他第一次接触计算机，是因为要用Fortran语言在打孔卡上编写一段指令。“怎么能让编程这件事更简单一些呢？这件事一直让我很着迷，我对自己说，好吧，我很想做这件事儿。”里夫斯说道。

于是里夫斯又主修了计算机科学，一直住在南帕萨迪纳，每天都要开车去波莫纳上学。一个偶然的机会改变了他的人生。一次上学途中，他注意到学校的约翰·罗尔教授正坐在车站等公共汽车。里夫斯那时正在选修罗尔的课，他觉着如果罗尔继续在这里等下去，肯定就不能按时到校讲课了。于是他邀请罗尔搭他的车。在一路上的交谈过程中，罗尔讲起了他在JPL深空探测任务中所做的软件开发工作。

后来罗尔成了里夫斯的导师，他安排里夫斯到JPL工作了一个夏天。到1983年里夫斯大学毕业后，这份临时工作又变成了他的正式工作。他的任务是为“麦哲伦号”宇宙飞船的测试设备编写程序。宇宙飞船以葡萄牙探险家麦哲伦的名字命名，它的任务是绘制金星表面的地图。

那时的飞船测试设备要充当两种角色。第一，它要充当地面控制设备，即一部与飞船间保持通信的电子设备；第二，它还要充当飞船本身，甚至还要包括正式飞船尚未装备的部件。这样，在总装完成之前的很长时间里，飞船的开发团队都可利用测试设备的相关功能，开展与飞船有关的安装、测试以及模拟飞行等工作。

里夫斯的老板罗伯特·安德森打破传统，用许多互联的单板机来替代基于大型主机系统的飞船测试设备。现在看来，是微型处理器的出现，顺利成章地促成了这样的选择。微型处理器简化了系统的结构：多个独立的板卡降低了系统整体

瘫痪的可能性，也排除了单一主机系统可能存在的瓶颈。里夫斯视安德森为最有权威的导师，他回忆说："他教导我，有时候你必须退回原地，重新从一张白纸开始。你原有的积累和经验终究会到达一个极限，让你无法从中找到突破。"

"麦哲伦号"任务之后，里夫斯短暂离开了JPL一段时间。就像在很多机构中一样，离开再回来，往往是为了创造升职和加薪的机会。1991年，重新归队的里夫斯成为了团队的主管，为"卡西尼号"宇宙飞船的测试设备开发软件。"卡西尼号"的任务是探测土星环以及土星的卫星。由于"卡西尼号"比"麦哲伦号"携带了更多的装备，因此里夫斯扩展了安德森的分布式架构，使其拥有更多的接口，并且可以模拟更多的需求。飞船上的每个设备都配有相应的专用计算机、软件及专门的硬件接口。大约20台计算机协同工作，并保持时间上的同步。它们记录数据，自动进行一致性测试，并保持与地面人工系统之间的连接。

"卡西尼号"于1997年10月成功发射。在此后的任务过程中，它于2004年6月30日进入土星轨道，2005年1月，其搭载的主要设备——由欧洲太空总署建造的"惠更斯号"探测器，飞向了土星的卫星土卫六（泰坦），并在着陆前报告了卫星大气层的组成成分。

20世纪90年代，NASA还发起了一系列机器人探测火星计划。说来有些不可思议，对于我们这样的没有翅膀的两足动物来说，陆地行走怎么可能比飞行还要困难呢？试想，在着陆时，首先要面临的是机械方面的问题，就是保证设备在着陆时不被撞坏；其次，一旦着陆，硬件设备要经受尘埃与极端温度的考验；再有，就是与地球之间通信的困难。这些机器人探测计划显然需要一种全新的设计观念。由于里夫斯在之前项目上的成功，他被选为团队主管，负责在"火星探路者"任务中为航天器开发软件（即"飞行软件"）。

这些新的挑战似乎令人望而生畏，宇宙飞船需要完成三个独立的任务：成功

发射并飞往火星、在火星表面成功着陆，以及为火星探测车“索杰纳号”充当气象站及通信站。而所有这些任务的完成都取决于飞船能否在着陆时的致命撞击中保持完好，是否能以有限的电力经受住强大的辐射轰击。

飞船升空后，就没有人能够直接碰触或探测到里面的设备，甚至连电子通信也很不稳定。里夫斯解释说：“如果天线的指向稍稍偏离了地球的方向，你就将收听不到来自飞船的信号，因为它的传输功率仅相当于一个100瓦的灯泡。信号传输的延迟加剧了问题的严重性，因为每次信号往返都要花费好几分钟的时间。”所有这些可能存在的问题，都会导致任务的失败。近半数的火星任务都失败了，而且绝大多数的失败最终导致航天器的彻底丢失。许多航天器都配备了冗余的硬件，尽量周全地预估可能出现的问题和可能面临的失败。但人们并不能事先预估到所有的问题，没有一项火星任务是完美无缺的。

原则上，当问题出现时，通常会有两种做法：(1) 放弃任务；(2) 尝试应急方案。作为一名曾经的汽车修理工，里夫斯认为，在尽力预想可能存在的问题的同时，还应该制造应急工具来对付那些无法预计的问题，即他所谓“对未知的无知”的问题。当你要去抢救一台几百万美元的航天器时，这一原则就可能会派上大用场。这种修复先行的思想还能让我们从问题中吸取经验教训，并逐步积累我们的工程知识库。里夫斯说：“如果放弃，你将一无所获。而且由于无法预计故障发生的具体部位，所以你也不可能知道未来需要做哪些维修。”

大多数工程师都知道，有好的故障，也有坏的故障。如果产生故障的原因较为单一，并且你后来把它修复了，这就是一个好的故障；如果找不到故障的根本原因，还幻想着解决它，结果只能是束手无策。总之，故障产生的可能原因是多种多样的——可能是单件设备的生产工艺问题，也可能是严苛的环境、粉尘、辐射及天外来客等，诸如此类。里夫斯承认：“我们发射的飞船还不够多，不足以让

我们真正了解太空环境。我们拥有一些知识，但还要有几分猜想。”

迄今为止，故障还都是好的。的确，到今天为止，所有的故障还都只是发生在地球上。要了解故障的发生及修复过程，最好先了解一下故障检测装置。针对每一枚发射上天的飞船，JPL都会在地球上建造一个装置，相当于飞船的复制品。这样，就可以演练某些动作，并验证命令的执行结果。如果飞船的动作出现异常，那么地球上对应的装置就可以提供一个诊断及解决问题的平台。

飞船设计成定时发送科学数据及飞船的状态信息。例如，如果我们收到一条“系统正常”的基本状态信息，却没有同时收到科学数据，那么出问题的可能是科学仪器，而非整个系统。地面控制部分可以充当“主动故障检测”的角色，通过发布命令改变飞船的行为，尤其是可以上传修正软件或重新启动各种组件。在“维修”一台个人电脑时，我们经常用重启机器来解决问题；对待飞船上的电脑，也可以这样做。尽管如此，如同在地球上一样，重启并不能解决所有问题。

“火星探路者”由两部分组成：火星探测车（一个能在行星表面做实验的移动机器人）及着陆器，后者兼具通信站、图像处理及气象站的功能。里夫斯的工作主要针对着陆器。他的任务是开发软件，将飞船从地球送入火星轨道、执行着陆任务并在火星表面开展各种活动。探测车的操作是独立的，它的通信要经由着陆器转发给地球。

1997年7月4日，“火星探路者”在火星表面成功着陆，带着安全气囊快乐地跳跃着，这让全世界都为之欢呼。它开始搜集科学数据并执行任务。但几天后，着陆器的计算机开始反复地自动重启，停止了有效的工作。里夫斯受命寻找问题的原因。里夫斯说：“如果一台计算机就在眼前，你只要按住Ctrl+Alt+Delete键就可以重新启动它，这一点在太空也是一样的。困难的是，我们需要知道，这台机器在重新启动之前正在做什么。”

幸运的是，里夫斯的小组已经在系统中设计了一项功能，可以跟踪计算机在执行命令过程中所产生的状态信息。相关记录中包含了所有派发的消息：软件运行过程中所发生的重要事件，与这些事件有关的所有任务，以及任务开始执行的时间。负责调试的团队[①]断定，故障是由低优先级任务阻断了高优先级任务所引起的。这就像一股出租车流占据了十字路口，挡住了救护车前往医院的去路一样。

着陆器的操作系统由风河公司开发，系统允许高优先级任务终止正在执行的低优先级任务，除非低优先级任务锁定了某项资源，而这项资源恰好是高优先级任务所必需的。在这个案例中，这项资源是一个用于传递消息的队列，消息中包含的科学数据要提交给某个科学处理任务。里夫斯解释道："我们的第一个问题是，为什么低优先级任务会长时间锁定这项资源？这时地球上正常工作的试验台终于派上用场了。"他的团队可以在这个试验台上运行那个与飞船上完全相同的软件。经过几天时间，他们遇到了同样的问题。问题终于暴露了：优先级被倒置了（请参看附注栏关于"优先级倒置：三个任务和一把锁"的内容）。

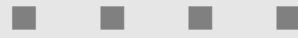
优先级倒置：三个任务和一把锁

优先级倒置源于三个任务与一把"锁"之间发生的资源争夺。低优先级任务（L）获得了这把"锁"并开始占用一项资源；高优先级任务（H）终止了正在占用资源运行的L，然后H开始运行；但当H要使用"锁"和资源的时候，却不得不终止，因为这时候，恰好有一项中优先级任务（M）开始运行，而M不需要这把"锁"，所以M不等L运行结束及锁释放就开始了运行。从最终结果上看，是M的运行阻止了H的运行，因此优先级被"倒置"了。*

---

① 系统调试团队的成员有瑞克·阿卡兹、戴夫·卡明斯、金姆·葛斯特罗、唐·迈耶、卡尔·施耐德、戴夫·史麦丝、史蒂夫·施托尔珀、格雷格·韦尔茨、帕姆·吉冈，他们都属于JPL，还有迈克·德里曼、布莱恩·拉扎尔及丽莎·史丹利，他们都来自于风河公司（Wind River）——"火星探路者"所用操作系统VxWorks的供应商。

优先级倒置的最佳解决方案，是由卡内基–梅隆大学的卢莎、拉格纳萨 · 拉杰库马尔及约翰 · P. 莱霍茨基提出的：当H发现自己需要L所持有的“锁”时，将L的优先级提升到与H同等水平，即让L“继承”H的优先级，综合的效果就是，让L的优先级高于M，直到L将“锁”释放掉。



* 我们的插画家艾丹 · 戴利建议用托尔金的小说《中土世界》来解释优先级倒置所带来的好处：
- 霍比特人（低优先级）拥有至尊魔戒及一把锁；
- 索伦（高优先级）没有至尊魔戒就不能复活并统治世界；
- 因此，人类（中优先级）可以和平统治世界。

解决这个问题看似简单：修改一下软件就行了。风河公司也提供了修改软件的简便方法，但如何在几百万英里之外进行修改呢？对于桌面电脑，安装一个修复程序实际上就是安装一张新的光盘，新版本软件的全部传送过程包括：先取得软件介质，再将其放入电脑，然后运行其中的安装程序。但是在“火星探路者”那里，里夫斯与他的团队要跨越上百万英里来修改软件，其传送的过程就充满了挑战：地球与飞船之间的传输速度只有每秒钟1200比特，或者更少。里夫斯要尽量避免上传整个软件，因为其中的绝大部分都不需要修改。所以，他只上传了新旧版本间有差异的那部分代码。

修改软件有点像心脏手术，当病人醒来前，你希望一切就位。对于“火星探路者”来说，里夫斯和他的团队已经预见到了软件可能需要修改，他们在设计上保留了两份软件的拷贝，一份用于正式运行，另一份留作备用。因此，他们可以将新上传的软件与飞船上已有的备用软件进行合并，然后存放在备用的存储空间上，并检查上传软件的完整性，在确认无误后，再将备用软件切换为运行状态。

为什么在测试阶段没能发现这一问题呢？一般首先想到的答案是，测试人员迫于时间的压力，每当碰到问题出现，就只把计算机重启了事。而如果更进一步

去想，就可能与组织本身的学习特点有关了，里夫斯这样来描述它：“建造飞船的过程，实际上也是学习如何操作的过程。只有当你能越来越熟练地操作它时，你才会交给它更多的任务，因为你已经熟知了它的工作方式及相关功能。而在早期的测试阶段，你其实还不敢放开手脚使用它。尽管你试图以实战的心态来完成测试，但很多时候，做得还远远不够。”而随着飞船的成功着陆，科学家们越来越有信心，他们开始更加充分地使用这台新设备了。

里夫斯的某些同事这样总结这次教训：“测其所飞，飞其所测。”也就是说，飞船在实际使用中所做的任何事情，不能超出其测试的范围。他们认为，问题的出现正是由于违反了这一原则。但实际上，这一原则只在理想情况下才有意义。在实战中，恶劣的环境，以及上面提到的科学家们不断增强的信心，都有可能使得“飞其所测”这个目标难以实现。反倒是另一个经验值得记取，就是“为不可预知的事情做好准备”。

2000年，里夫斯被指派参与火星探测车任务。在那里，他很快又得到了另一条经验：在火星上，硬件的寿命是有限的。环境温度会在白天的0℃与夜晚的零下100℃之间摇摆。夜间时，电池可以将设备的温度维持在零下40℃，但如此剧烈的变化会使硬件产生疲劳，使焊点断裂，并最终导致电气故障。探测车预计的寿命只有90天，所以不能浪费15天的时间来修理它。如果出了故障，里夫斯和他的团队就必须马上进行排除。

更迫切的是，里夫斯还必须为两部移动机器人编写通信软件。之所以用两部，是为了应付可能出现的失败。机器人的移动性使设计面临选择：是由地面控制机器人的动作，还是给机器人一定的自主权？用计算机科学的术语说，就是在高级控制与低级控制之间进行选择。地面控制应该向机器人发出什么样的指令呢？是“左前轮转六圈，右前轮转五圈”，还是说“把车从这儿开到那儿，小心躲避障碍”？

再或者是更简单地说“去找些奇特的东西来”？第一种选择肯定不能胜任科学研究的任务。

地面控制的另一个困难是通信。在控制过程中，地面需要保持与火星表面某点之间的持续通信，但问题在于这颗红色的星球在一刻不停地旋转着。当飞船在太空中时，地球的旋转可以保证总有一个地面站能看到它，并和它对话，使通信处于24小时全覆盖状态。但是，由于火星的旋转，探测器也随火星一同旋转，最终它的天线指向偏离了地球，那时，任何一个地面站都无法再与之通信。

那个“去找些奇特的东西来”的选择，要求机器人拥有很高的科学知识水平。机器人探测任务，多半是响应式的，即一系列的“发现”与“响应”交替进行。举例说，如果机器人在岩石中发现了罕见的化学物质，地面控制就会发出命令，让它去完成更多的实验。最近科学团队给“奇特的东西”做出了清晰的界定，但里夫斯相信这种界定今后还是会变化的。由于探测的距离超过了50个光时[①]，地面控制不可能让着陆器连续锁定一个指定的目标。里夫斯预言说：“机器人需要更多的自主性。”计算设备需要大量更科学的实用知识和技能来自我修复，这是里夫斯在JPL的同事阿德里安·斯托伊卡目前在研究的课题。

适应和发展

针对一个历时百年的任务，我们所面临的设计问题是：高能粒子对飞船内部敏感电子装置的撞击，以及极端温度所引起的机械损耗。目前对于这类辐射及温度变化问题，一般会采用主动屏蔽的方法：有点像高科技的保温瓶。但这种常规

① 光在50个小时里行进的距离，约为54亿公里。火星与地球最近距离是5400万公里，最远距离是4亿公里，平均距离1.2亿公里，50光时相当于54亿公里，为最近距离的100倍，最远距离的近14倍，故原文可能有误。——译者注

的方法会增加飞船重量，消耗更多能量，而重量和能量恰恰是太空任务中最宝贵的资源。

当然，人类同样无法忍受这样极端的环境。就像其他灵长类动物一样，我们最适宜的温度在22℃左右。就物种而言，人类在穿着保暖衣物，或避免强烈阳光直射的情况下，能够适应的温度范围是从－40℃到40℃。电子设备可以通过改变电路连接及电压，来适应温度的变化，避免“笨重暖水瓶”综合症。就像孩子知道在冷天穿上夹克一样，当察觉到环境变化，或某项特定任务无法完成时，电路也应该可以自我调整：这是阿德里安 · 斯托伊卡富有创见的研究提案。

斯托伊卡于1962年出生在雅西：摩尔达维亚（英语为Moldavia）公国首府。摩尔达维亚与特兰西瓦尼亚、瓦拉几亚共同构成了现代的罗马尼亚，但在第二次世界大战之后，摩尔达维亚的一部分被划归苏联。长久以来，雅西都被视作罗马尼亚的文化中心，这里有罗马尼亚最古老的大学，斯托伊卡的父亲及兄弟就在这所大学里教经济学，他的母亲则在国家剧院做会计师。

雅西的学校主张严格的数学及科学教育。“这对孩子来说是非常艰难的。”斯托伊卡说。但他很感激在那里所受的训练，比起国内其他地区的孩子，他相信这些训练给了他很大的优势。罗马尼亚中学的理科教育采纳了美国和俄罗斯最优秀的数学及物理教材，学生们能接触到西方的译著，如《费曼物理学讲义》及伯克利的课程讲义。1981年，当斯托伊卡到雅西技术大学读书时，他已经打好了坚实的基础。很快，他开始热衷于电子及计算。作为一名大学新生，他开始用打孔卡编写程序；到毕业时，他已经可以自己动手建造微处理器系统了。

在互联网出现之前，罗马尼亚的图书馆里很少见到先进的技术文献。尽管有一些数学及科学的书籍，但有关技术方面的西方书籍很匮乏，因此学生们一旦获得某些资料，就一定要拿来复印并相互分享。罗马尼亚的大学里有许多外国留学

生，电子元件可以从他们手里买到。斯托伊卡说：“我们到黑市上买这些东西，拿来做大学里的项目。”

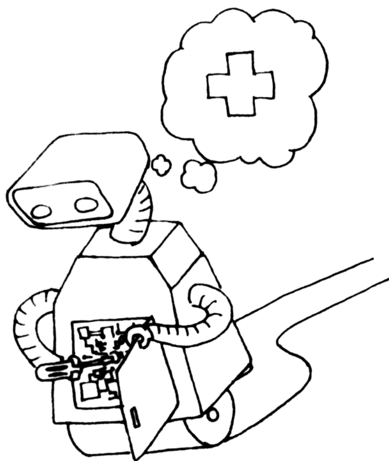
闲暇时间里，斯托伊卡喜欢读科幻小说，他最喜欢的作者是艾萨克·阿西莫夫、史坦尼斯劳·莱姆以及其他一些欧洲的科幻小说家，觉得他们富有灵感。斯托伊卡回忆说：“科幻小说启发了人的头脑，让人充满梦想。后来，在生活中我听到过大致这样的说法：‘如果你在脑子里见过它，就能在现实生活里实现它。’”受到科幻小说及电影（如《银翼杀手》）的鼓舞，斯托伊卡想象着在一间小实验室里，或公司里，自己能亲手制造人形机器人。1986年他从雅西技术大学毕业，获得了电子工程学硕士学位，之后加入了一家生产传感器及测量设备的公司。他还记得自己跟同事们说：“伙计们，将来有一天我会为NASA工作。”斯托伊卡承认，当时的每个人都笑了。“但我是认真的，我说不好，那是出于某种预感，还是想用某种方式来表达自己的梦想，但总会有那么一天，我会在某个技术最领先的地方，做出我的贡献。”

斯托伊卡的NASA梦想被推迟了，他要到澳大利亚继续完成他的学业。在墨尔本的维多利亚大学，他的博士研究工作包括训练机器人，通过让它观察示范者的动作来学会移动。1996年，在斯托伊卡完成论文的同时，他赢得了“绿卡乐透”——美国政府的一项多样性移民签证计划，这使他获准移民美国。为了满足移民要求，他抛开一切来到美国，加入了NASA——他梦想中的雇主，并开始在喷气推进实验室工作，一直到现在。

斯托伊卡的第一份工作是制作能适应特殊环境的电子器件，如在其他行星表面，或者地球上的极端环境，如火山口、核反应堆及油井的钻孔中。人们一般都采用常规的“暖水瓶”方法来保护电子器件，使其与外界环境隔离开来。但斯托伊卡想到了第二种可能性：当环境变化时，可通过电子器件的自我重新配置，来

替换掉损坏的部件。

斯托伊卡注意到六次任务——“旅行者1号”、“旅行者2号”、“海盗1号”、“海盗2号”、“伽利略号”及“麦哲伦号”——在发射后都出现了问题，而备用零件在这些任务的拯救过程中，都起到了决定性作用。备用的或者冗余的零件显然是有用的，但它们会占用空间及重量，这就意味着要压缩科研仪器的空间。而且重量越大，也就意味着成本越高（大约1万美元/磅）。人们为了以防万一而只好给飞船配置充足的备件，最后只能是无限制地推高成本，因为设计者预先不可能知道多少备件才算充足。于是斯托伊卡想换一种方法，即通过制造“可自我重新配置”的部件来满足可靠性要求。这个方法使得电子器件具有更大的灵活性和通用性，但是需要有对故障的精确定位及修复的能力。



人类的身体提供了好的例证：皮肤伤口通常几天就愈合了（通过邻近的皮肤再生），断臂需要几周的时间复合，而截肢则需要一个假体来修复。如果飞船的设计能让小故障得以就地解决，严重的故障则通过更换备件来解决，那么飞船就能凭借少量的备件而存活更长时间。

斯托伊卡认为他可以利用进化思想来实现这样的适应性。当环境平静而温和

时，定制化的设计容易获得成功；但是当周遭环境变动不羁时，只有善于应变者才能生存。这样的事实比比皆是：被抗生素攻击的细菌，环境压力下生长的植物，还有，在动态技术领域里探索的人们。

斯托伊卡听说过发明家约翰·科扎（斯坦福大学的顾问教授）所从事的工作。科扎采纳了他的导师约翰·霍兰德的遗传算法思想，将其应用于电路设计。在科扎的框架中，遗传算法的应用过程是：首先，从一个小的集合开始，集合中包含了一些电阻器、晶体管及电容器；然后，具体说明要解决的问题，例如，设计一个立体声调谐器；最后，算法将自动开发出一套有效的电路，解决这一问题。这一过程的灵感来自于生物的进化，算法首先从众多的满足要求的电路中进行筛选，从而获得若干个最符合要求的电路；然后组合这些电路的"遗传密码"，产生出新的子代电路；从子代中再次筛选出最优秀的电路（在立体声调谐器的例子中，指的是那些能将音乐调制到最完美的电路），以此类推。

不过，斯托伊卡的目标与科扎不同。科扎利用进化原理来设计专用电路；斯托伊卡则是要设计并制作出柔性电路，一旦电路付诸使用，它可以借助遗传的方法进行重新配置，以适应不同的环境。他说："我们的思路是实时进化。"斯托伊卡要筛选出一种能独立进行自我调整的芯片，这样当辐射及温度发生变化时，就能使设备的某些特性也随之改变。通过重新设计就可以找到不同的优化电路拓扑，或不同的偏压设置，以使设备在新的环境下获得最优的性能。

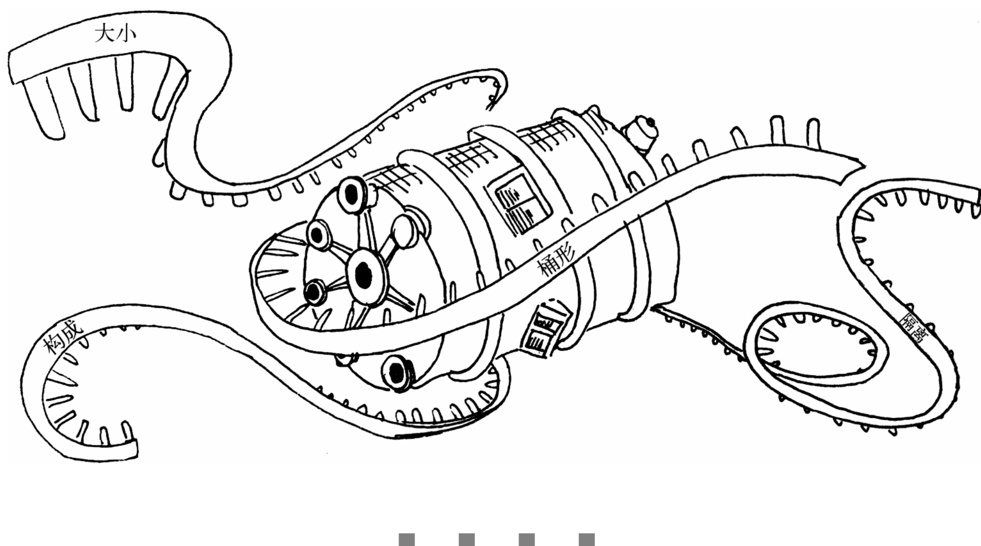
斯托伊卡使用的硬件有一个专业的名称，"现场可编程晶体管阵列"，它具有一个二维的、曼哈顿式[①]的网格结构，由晶体管及其他电子元件组成，各种元件之间通过晶体管开关相互连接。这些晶体管开关有两种状态：关闭（二进制的值为1），

① 曼哈顿式：指一种城市建设模式，有高密度的高大建筑，同时有完善的交通规划，有多种交通工具可供选择。——译者注

此时允许电流通过；断开（二进制的值为0），此时阻止电流通过。开关的状态由比特串（由若干0和1混和而成的数字序列）来控制，这个比特串则被定义为电路的“遗传密码”。通过一次进化搜索，可以对各种组合的比特串进行测试、比较，从而帮助改进电路的性能。斯托伊卡已经可以展示他的电路对温度变化的适应，其变化范围可以从零下180℃到零上120℃。

斯托伊卡设想了一种多级适应模式：有些小范围的闭合回路，相当于系统在细胞级别上的适应（就像皮肤修复伤口）；如果有更多的“细胞”，那么整个细胞群都将表现出自组织的特征，这将形成更高级别的协同与进化。适应性强的部件具有更强的生存能力，但这是否会造成危险并导致失去控制呢？斯托伊卡的这种担心源于他少年时代所热爱的科幻小说中的描述：“如果将来某个时刻，我们把这些人工系统与我们的生命系统相结合，或者允许它们自我复制，那么，一旦它们开始胡闹，想要阻止它们可不只是拔掉电源那样简单，它们会因为数量巨大并能够自我复制，而摆脱我们的控制。”尽管如此，他仍然认为其正面意义要远大于可能的负面因素。

在展望太空探索的未来时，斯托伊卡提到了“行星地球化”的理念，例如，在火星上创造出人类宜居的环境。“那是一个从功能出发而构建的生命和人工的混合系统。”他不无幽默地调侃，或许有一天，当人类降落到“地球化”的火星时，那些机器人会组成一个欢迎小组，并开启香槟。它们会干杯吗？



■ ■ ■ ■

遗传算法是一个工具，在设计系统时，它能突破人类思维的惯性，取得出人意料的结果。

——路易斯·奎尔斯

| 第 3 章 |

路易斯·奎尔斯

为设计团队注入进化思想

二次世界大战期间，在田纳西州的山里，诞生了一处秘密所在：橡树岭国家实验室。最初，它是作为曼哈顿计划的一部分，奉命生产和提纯用于制造原子弹的钚。从那时起，实验室的研究就是多元化的，涉及能源、系统生物学以及材料科学等多个领域，并在核物理及工程领域，保持着世界级的领先地位。即使在财政紧缩的年代，美国能源部依旧以资金和大型工程项目委托的方式，向橡树岭的科学家们提供慷慨的资助。

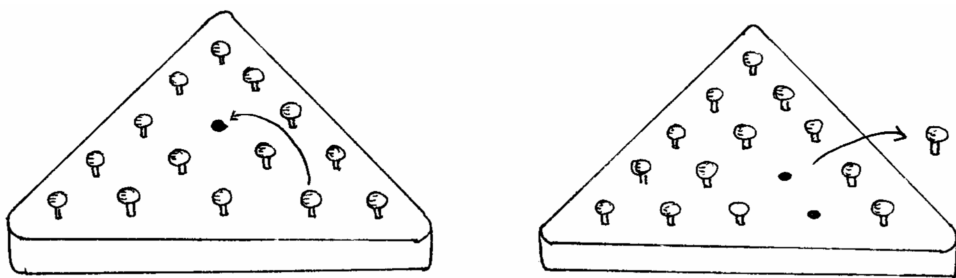
路易斯·奎尔斯是橡树岭的一名高级研究员及核能工程师，他称自己为“系统集成者”。为了实现核反应堆的复杂工程设计，他常常把各种不同的零部件组合到一起。系统集成是心理学与工程学的结合，它需要说服负责设计零部件的专家们改变他们的理想化设计思路，来满足整体框架的需要。系统设计师往往会像艺术家那样，对自己的作品产生依恋之情，而进化计算可以帮助工程师们克服这种设计上的迷恋倾向。

奎尔斯生于1963年，在田纳西州的孟菲斯市长大。他的母亲来自田纳西州西部乡村的一座农庄，他的外公是一位富有创造力的多面手：集焊接工、机修工与机械师于一身。奎尔斯解释说：“我今天做的很多事情其实跟他差不多，如果想弄

清楚一个东西是怎么工作的，我就把它拆开，然后再想办法重新组装起来。” 奎尔斯的父亲是一名商人，他不断地向儿子灌输一种强烈的职业精神。奎尔斯回忆说：“他说90%的成功来自于按时上班、着装得体以及努力工作。” 他牢记父亲的教诲，并常常对自己能够有幸加入橡树岭而表现出一种感激之情。

奎尔斯进入了孟菲斯市的一所公立学校，之后又加入哈丁基督教中学：一所私人教会学校。他用“中等生”来形容自己的学生时代。奎尔斯对工业美术感兴趣，其中包括了建筑学、工程学及物理学。1981年，奎尔斯开始在田纳西大学诺克斯维尔分校学习建筑学。不久，他发现自己的绘图水平不怎么好，而且他对建筑内在结构的兴趣远远大于对其外在样式的兴趣。他开始考虑去修一个工程学学位，问题是选择哪个专业——机械工程、电子工程、土木工程，或者是航空航天工程，有许多专业可供选择。奎尔斯决定去面见各个系的主管。在与核工程系的系主任皮特·帕斯夸交谈后，对方劝他选择核工程专业。奎尔斯回忆说：“他向我灌输核工程的重大意义，说那会对人的一生产生深远的影响。” 核工程的另一个好处是这个领域中的从业人员极少。每年其他工程专业的毕业生有200~300个，而与核有关的毕业生一共就只有大约20个。奎尔斯觉得，如果他答应去帕斯夸的系，至少会有一个人在乎他的成功。

核工程对于喜欢搞小发明的奎尔斯来说非常有吸引力，课程包括数学、计算、工程学、热工水力学、机械学、泵及涡轮机等。在一年级的计算课上，他对一个简单智力题的解法预示了他后来的设计风格。课堂作业是写一段程序来解开“楔子谜题”（peg puzzle）：这种游戏经常出现在美国南部的饼干桶餐厅的桌面上。游戏开始时，棋盘上有15个洞，洞中放着14个楔子，因此有一个洞是空的。玩家每次将一个楔子跳到空洞中，中间必须且只能隔一个楔子，每跳一次后，被跳过的楔子要被拿掉。解题的要求是，最后棋盘上只能剩下一个楔子，而且要放在它最初的位置上。



游戏中，每跳一次就要拿掉一个楔子

为了这个作业，奎尔斯使用了具有回溯机制的Prolog语言（见附注栏内容“不断尝试：回溯法基础知识”）写了一段程序。奎尔斯回忆说：“这点真的很让人赞叹，计算机其实并不聪明，但它永不放弃。你只要做好设置，让它按照一定步骤一直试下去，最后它总能找到答案。”他也承认他给出的解法并不算简洁，但不是所有的问题都适合用简洁的方法去解决。他所获得的初步经验是：只要你能提出正确的问题，计算机就一定能够绞尽脑汁地找出答案。

不断尝试：回溯法基础知识

回溯是一种技术方法，适合于求解楔子谜题、数独或其他相似类型的解谜游戏。在楔子谜题中，回溯方法主要体现为如下过程：设棋盘当前所处的状态为S，选择一个可跳的楔子以及跳向的位置，并完成跳的动作，就会产生一个新的状态S'。继续这一过程，直到处于无法再跳的状态。如果最后的状态是赢，那你就成功了，否则就要回到最后状态的前一个状态，然后选择一个不同于此前的跳法，再跳。

回溯是一个失败之后回退的过程，即恢复到原来状态，进行新的尝试。为了使这个过程有条不紊地进行，程序必须记住之前访问过的各个状态，也就是程序所尝试过的每一跳的始发状态。在奎尔斯所使用的Prolog语言里，程序员可以设定一个目标状态，这样编译器就可以直接让硬件去做回溯求解的工作。

1984年，也就是在三里岛核事故发生五年后，奎尔斯开始做他的硕士论文，主题是核反应堆的信号校验：测定不同传感器之间的通信差异，以及通信结果的可信性。在三里岛核事故中，反应堆芯出现了部分熔化，只是因为反应堆容器和外壳结构的完整性，才避免了辐射的大面积扩散。专家们对事故的分析揭示出，正是由于操作人员对于传感器读数（其中有些是错误的）的草率推断，才造成了其后的决策失误。

如果控制台上两个传感器的读数不一致，你会相信哪一个呢？奎尔斯研发了一个信号校验模型，可以作为核电厂操作人员的助手，在后台持续运行。如果温度升高或压力增大，它会提示操作人员。奎尔斯说："这是一个顾问程序，它告诉操作人员，哪些信息是可以忽略的，哪些信息更有可能是真实的，以便于他们做出正确的决策。"

为了实现顾问程序的告知功能，奎尔斯使用了各种工程类的经验法则，例如，出自同一家生产商，而且由同一名工程师所标定的传感器，会给出相近的读数。如果不是这样，出现了异常的读数，那么偏离了基准范围的读数有可能就是错误的。历史数据也可以提供帮助。当某个泵在运行时，如果容器A的压力值一直都高于容器B，那么一旦压力传感器的显示出现反常情况，就可能是泵出了问题。在三里岛核事故中，人们误以为一个阀门已经关闭了，但实际上却没有。奎尔斯的程序可以帮助操作员排查出此类问题。

奎尔斯认为人类操作员仍然具有很重要的作用，但计算机的支持可以起到很好的辅助作用。与人类操作员不同，计算监视设备永远不会注意力涣散，也不会参加整夜的聚会之后打瞌睡。奎尔斯说："它们所做的全部工作就是一直专注地盯着某些东西看。而且通过编程，可以让它具有学习能力：持续地学习（通过建立事件数据库）什么是正常，什么是不正常。"

三里岛时代的核电行业还无法使用顾问程序，但现在它们都已经用上了。人们开始发现，在许多领域中，都存在着同样易被破坏的“人机平衡”。计算机的优势在于能对常态的信息进行监视，一旦发生变化，能够及时提醒操作人员。而人类则能够对更广范围的信息进行综合判断，并做出决策。在时间紧迫的危急关头，当同时有大量信息需要处理时，人类需要有计算机的辅助。有些组织内部经常使用检查清单和专家系统来协助人们进行危机处理。在核电厂里，一旦发生故障，检查清单可以指导操作员保护反应堆芯的完整性。

奎尔斯于1988年完成了他在田纳西大学核工程专业的硕士学业，并于1991年进入橡树岭国家实验室做他的博士论文，此后便一直留在那里。为了完成论文，奎尔斯开始研究一种叫“仿星器”（stellarator）的核聚变装置（核聚变是恒星的能量来源，stella是拉丁语中“星”的叫法）。奎尔斯的工作是建造一个弹丸发射器，用于向仿星器内的环形等离子体注射燃料。

没有学校能为需要许多部门和人员共同参与的大型工程项目培养现成的人才。经过一个项目就会让一个年轻的设计工程师领略到项目能够成功的必备要素。直到现在，奎尔斯还在赞叹那些橡树岭的员工，称他们让一个初出茅庐的科学家获益匪浅：“在你愿意与之共事的人当中，他们是最聪明的。但是他们不会跟你夸耀自己的聪明，也从不自以为聪明。他们只是一些思维敏捷的人，每天上班，做他们该做的。但如果你倾听他们的谈话，一定会学到很多。”

橡树岭的事业具有深远的影响。与橡树岭毗邻的Y-12国家安全联合体负责研制核武器组件，在他们连续不断的核废料清理中，在他们殚精竭虑的持续努力中，原子弹发展的历史痕迹依然清晰可见。从20世纪70年代开始，橡树岭划归能源部管辖，它旗下的4300名员工开始着手应对“重大挑战”问题，涵盖了从中子科学到纳米材料设计等诸多领域。橡树岭的占地面积扩大到33 750英亩，这里的设施

为世界顶尖科学家间的交流以及科学实验研究提供了方便。

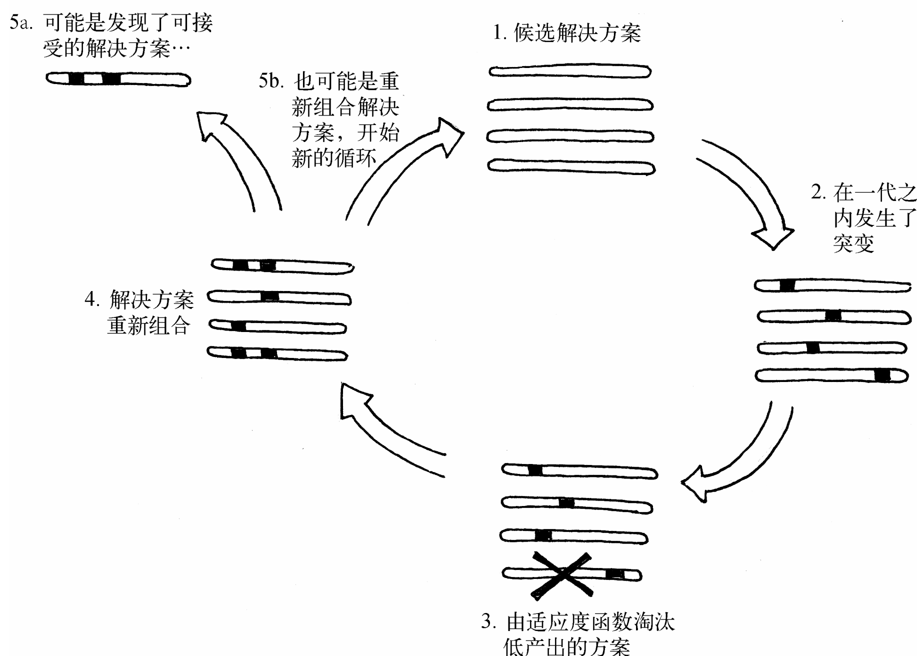
奎尔斯非常适应这里开阔的学术视野。他善于将不同的组件攒在一起，使其发挥出特殊的功效，来适应各种不同的需求，而他本人也为此而备受赞誉。例如，在一项前瞻性的工作中，NASA计划在月球上建造基于核动力的人类栖息环境。奎尔斯的工作涉及很多相关因素：燃料、核芯、安全壳、防护罩、能量转换设备以及电子设备等。在以上的任何一个方面，他都称不上是专家，但是在实施这个项目时，他开始向每个领域的专家请教。他把专家们一起请到会议室，让他们发表自己的见解。奎尔斯如此描述道："在那个时间，那个房间，我找到了所有的办法，找到了我的系统。我把它们合在一起，如果达不到理想的效果，我可以马上修改。"

像许多人一样，奎尔斯的设计会从一个初步的构思开始，然后进行逐步调整。但是如何甄选出好的构思呢？而且，一旦有了构思，如何对它进行调整呢？奎尔斯的方法是使用遗传算法，先从几百个随机的初始设计开始，而调整的过程则涉及对改进方案的搜索。例如，能否提高燃料系统的效率？提高效率的同时，是否会增加少量成本？有时，调整并不能改进总体性能，但是却为改进提供了新的可能性。例如，改变泵的材料并不会改变泵的重量，却能增加它的流量并提高能效。理想的调整策略不仅包括了搜索改进方案，有时也会关注一些中性的甚至是负面的改变，看看这些改变能否为今后的改进带来更多的可能性。除了分别调整每一个设计，遗传算法也允许将不同的设计相互结合。

遗传算法（或进化计算）策略在应用于设计时有以下过程：(1) 从许多初始设计开始；(2) 随机改变各个独立的设计；(3) 将两个优秀的设计相结合，以产生更好的设计。

在开始采用遗传算法之前，奎尔斯需要花三周时间进行草案设计。他承认说：

“我的确惹恼了大家，因为我的设计完全背离了他们的初衷。” 但他又说：“在计算机上，我只要改变一下约束条件，然后按一下按钮就可以走开不管了。”遗传算法还可以把这种“按下按钮就走开”的好处分享给其他计算机设计方法，如回溯法（见上文附注栏内容“不断尝试：回溯法基础知识”）。设计者只要在计算机模型中把约束条件进行适当的编码，当增加新的约束条件时，只需要重新运行这个模型就可以了。与人类设计者相比，遗传算法模型不会对既往设计产生情感依赖，也就是说，它们不会刻意忽略对原有设计的修改。它会从一个全新的初始设计方案开始，并有可能得出全然不同的结果。



遗传算法的基本循环。(1) 从一套候选的设计开始，把其中的每种设计当做一条由基因序列组成的染色体。每个“基因”代表一个特定的设计选择（如反应堆容器的形状）。(2) 遗传算法通过突变的方式，在每一条染色体上随机改变几个基因。(3) 算法对染色体进行评估，淘汰那些得分低的染色体。(4) 留下来的染色体被重新组合（但是要记住其中得分最高的，以免找不到更好的染色体）。（5a）如果找到了令人满意的染色体，那么结束循环。（5b）否则，开始新的循环

当新的技术出现时，人们倾向于保持他们原有的偏好。如果怀疑这一点，你可以回想一下，在20世纪初的很多年里，汽车都保持着“无马四轮马车”的样子。再想想近些年来在商业以及技术上的创新——特快专递、微处理器、互联网以及手机，这些都是由传统企业以外的人为了完善现有服务而发明的，而企业内部的人则往往受制于原有设计的思维框架而不能突破。

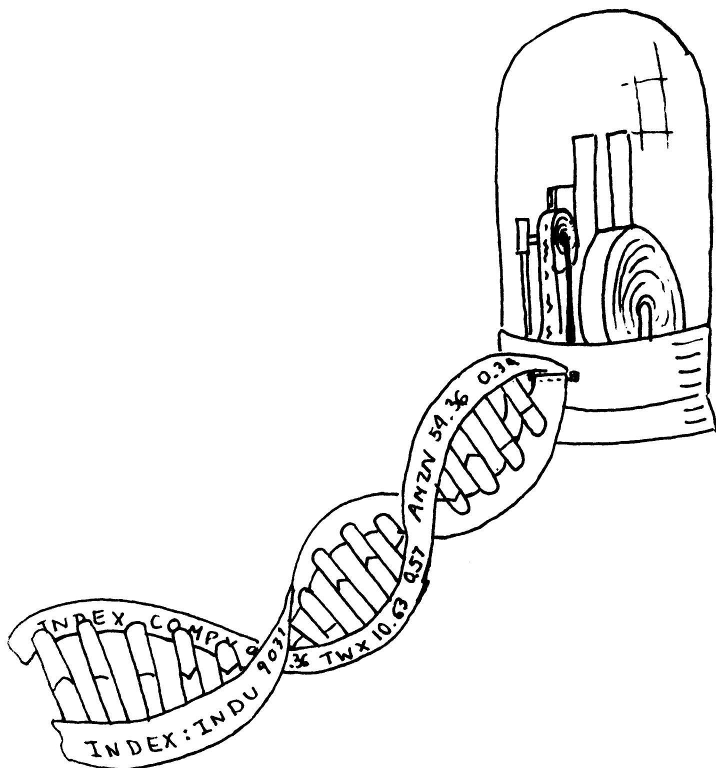
奎尔斯回想起为宇宙飞船设计容器的经历。容器必须能够屏蔽来自太空的伽马射线。问题的重点是把屏蔽材料添加在什么位置，在不同材料表面增加涂层的方法有数千种。奎尔斯说：“遗传算法给出的方案是，把屏蔽材料做成一个薄层，夹在容器材料的中间，这样可以使成本最低，同时重量最轻。” 屏蔽专家认可了这个设计的正确性。遗传算法只用了几分钟的时间，就对预选设计进行了几千代的筛选计算，并给出了解决方案，节省了时间和金钱。

遗传算法可以同时处理很多约束条件：远远超过人工的处理量。例如，在建造核电厂时，设计者要考虑核废料的问题。处理核废料是一个极其复杂的问题，涉及资金、人力及诸多风险。 奎尔斯预言：“遗传算法可以帮助我们在更大的设计空间里，利用那些更大、更细致的模型，对设计做出优化，并找出最优的解决方案。”

遗传算法还可以加强项目中的团队协作。在过去，当奎尔斯针对一个设计要点给出折中方案时，许多专家不喜欢他的结果。但是，当遗传算法筛选出100个设计方案后，专家们才看出折中的意图所在，他们的态度也因此有了改善。奎尔斯说：“遗传算法帮助各个子系统的专家们理解了其他子系统的约束条件。” 于是，团队成员之间有了更多的合作。

尽管奎尔斯对遗传算法充满信心，但他更加强调人工设计的重要性。他们可以检验计算机生成的设计方案的效果。实际上，设计者向计算机展示了一个包含

众多可能的设计图景，并允许计算机对它们进行优化。因此，人才是主角。“如果有了飞机的遗传算法模型，那么遗传算法可以优化这个模型，” 奎尔斯说，“但是它无法在飞机与直升机之间做出选择，除非同时具备了这两者的模型。”





股市的行情时涨时跌，看不出有什么显而易见的原因。你一直在盯着大盘看，有些行情会重复，至少有部分行情会重复出现。时间序列表现得越完美，重复出现的可能性就越大。一年后，微软的股价可能翻倍，也可能对折，但下一个毫秒，却只可能有1美分的涨跌。

——杰克·洛夫莱斯

| 第 4 章 |

杰克·洛夫莱斯和阿穆特·巴拉姆比

追赶大潮

几年前，在一张“动手造飞机”的节日海报上，出现了这样的宣传语：“制造它靠的是大脑，但驾驶它靠的是身体。”用这句话来描述金融市场上的交易行为，一点儿都不为过。你查看历史数据，并试图发现一种模式。你打算按照这个模式来操作，却永远无法知道接下来会发生什么。于是你只能冒险地依照某个特定策略而孤注一掷。你的头脑完全被直觉、本能或者其他因素所支配。假设一种模式在关键时刻没起作用，而你却投入了一千万美元来押注，那么你该如何收场呢？即便那不是你自己的钱，这个结果也会直接影响到你的职业和声誉。

*程序化交易*兴起于20世纪80年代，指的是用计算机来完成大宗股票交易。较新的方向是研究短线交易，也称为*微观市场交易*，在不需要人工干预的前提下，借助于自适应的规则来完成交易。进化计算可以发现新的规则，并能在条件发生变化时相应地改变规则。这正是一位冲浪高手与另一位前网球明星的得意之作。

1980年，杰克·洛夫莱斯出生在迈阿密附近，童年生活在佛罗里达州的博卡拉顿，由同为教师的母亲和继父抚养大。母亲的专业是数学，杰克记得，那时为

了测试他的思维敏捷度，母亲养成了一个“坏毛病”，就是先在一张纸上给他讲解二次方程题，然后把纸撕碎，并要求他按照她的方法重新做一遍。他回忆说：“你必须完全理解她的方法，否则就得自己来推导。”

在高中时期，杰克喜欢化学和数学，但最终他决定去佛罗里达大学读哲学。他对自己的未来没有什么明确的打算，但有一项活动让他很感兴趣，那就是冲浪。他也曾遐想成为一名教授。他旁听了许多数学课，发现微积分很无聊，但却很喜欢混沌理论。他的学习成绩不算太好，他说：“我的统计学和C语言都不及格，而且是两次不及格，那是我学过的唯一一门编程语言，所以第二年，我转去学潜水了。”

在丢盔卸甲地离开佛罗里达大学之后，洛夫莱斯考虑换一个地方生活。他来到弗吉尼亚，到他父亲身边，并在弗雷德里克斯堡的一家小型搬家公司的仓库找到一份工作。他的工作是将六块胶合板钉在一起：是做箱子，而不是解二次方程。不久他就被提升去运送家具。

洛夫莱斯对搬家公司业务的运作方式很感兴趣，他想找到盈利的最佳途径。他认识到，货运业务的利润空间很小，利润取决于行程的安排。于是他写好了一份业务计划，并借来一套西服。当公司的CEO来工作现场巡视时，他安排了一次会议，并见到了她。CEO非常认可他的业务计划，并让他担任公司的第一个技术总监，给他加了薪，在原来每小时4.25美元的基础上，提高了一大截。

洛夫莱斯的目标是保持卡车的满载。软件是用Cobol语言（最古老的编程语言之一）编写的，它依据的事实是：如果货车是从弗吉尼亚开往加州，那么先送一批货到田纳西，卸货后再装一批货到加州，会比空车开到田纳西再装车前往加州赚到更多的钱。这是一个业务优化的典型案例。洛夫莱斯已经实现了用更少的卡车，通过更高效的路径选择，来提升业务的收益，这对于2000年处于技术落后状

态的运输行业而言，提升的效果尤为明显。洛夫莱斯说："我们所说的毕竟是运输行业，这里可没有多少数学家。"

洛夫莱斯加入了美国运输与仓储协会，并继续开发涉及多公司的货运软件。随后他又接到了亚壁（Appian）公司的电话。亚壁公司是一家位于弗吉尼亚州维也纳市的技术公司，它承担着一项超大型的工程任务，即为美国海军设计数据库系统。公司的技术优势在于速度和扩展性方面。亚壁公司的数据库程序基于一种叫做K的编程语言（参见下文附注栏内容"APL语言、K语言以及计算机科学教育宣言"）。虽然连简单得多的C语言都没能通过考试，但洛夫莱斯很快就领略到了K语言的威力，更是被K语言程序员的口头禅所打动："我不喜欢写太多代码。"

APL语言、K语言以及计算机科学教育宣言

1955年，计算领域的先驱霍华德・艾肯说服哈佛大学的管理部门，首次引入了"自动数据处理"——也就是计算机科学的硕士项目。艾肯雇用了一位年轻的加拿大研究生肯・艾弗森来教授一部分课程。艾弗森发现当时的数学符号不够严密，于是他开始开发自己的符号。他借用了线性代数中矢量及矩阵的操作符，并鼓励程序员思考用信息的集合来替代单一的信息单元。1979年艾弗森荣获了计算机科学领域的诺贝尔奖——图灵奖，而许多语言的设计者都获益于他的APL语言。

尽管APL语言获得如此殊荣，但它却从未成为一种主流语言，并且几乎被大学教育所忽视。首先，APL语言从字面上看像希腊文：用的都是希腊字母。用APL语言写程序必须使用一种特殊的键盘，或者使用重新映射的键盘，而使用者要记住大量的字符映射关系。其次，这个语言使用的运算符功能异常强大，以至于让那些传统的程序员望而生畏。APL语言仅需敲打几次键盘就可以处理一百万乘一百万的矩阵，但传统程序员通常一次敲击键盘只能操作一个数字。

在20世纪90年代早期，阿瑟·惠特尼在APL语言的基础上，设计并实现了K语言，他也是另一个称作J的语言的发明人。K语言采用普通键盘，因此它的符号看起来并不可怕，但是对某些符号做了非常规的定义，如，“+”表示“加法”，但“+/”表示“对列表中所有项求和”。惠特尼通过使用巧妙的数据布局设计，执行各种效率的排序算法（为浮点数与小值域数分别设置算法），并坚持使用简单的实现，从而使K语言具有极高的执行效率。

实际上惠特尼扩展了APL语言操作符的威力，结果使得K语言可以处理从通信到数据库再到线性代数中的所有问题。通常要求整合Perl、Java及SQL等多种语言的应用程序，这里只要有一种语言就够了（本书作者之一萨莎就使用K语言做他的所有工作，我们不能否认自己对这门语言有偏爱）。

在写本书时，基于Pascal、C，有时是LISP的语言（如C++及Java）占据了计算机科学领域的主导地位。数据量大的学科（如金融数学与生物学）中则教授Matlab及R这一类的向量语言。

由于多核芯片的出现，计算机科学的教学必须做出相应的改变。多核芯片中内置了多个处理单元，每个单元都能单独运行一个程序。为多核芯片编写一个顺序执行的程序，意味着只用一个核来做所有工作。程序员也可以使用像Java这样的语言编写多线程的程序，来利用所有的核，但大多数程序员的技术能力有限，而一个漏洞百出的应用程序甚至会轻易导致生命及财产的损失。我们迫切需要这样一门语言，首先它支持多核，其次它具有一种特殊运算符，可以将矩阵和数据库表当做单个的数据单元来处理。为此我们发布宣言：我们需要向学生教授具有超强能力的语言。

2003年，在亚壁公司工作了两年后，洛夫莱斯和他的同事合伙开了一家自己的公司。他们找到一间车库，继续为国防部提供咨询。当公司被收购时，他又回到了佛罗里达，一边冲浪（在海里，而不是互联网或别的什么），一边为像微软、

红帽子一类的公司提供咨询。

在咨询过程中，他结识了一位名叫大卫・雅各布森的海军军官，后者就职于海军医学信息中心。雅各布森的工作是流行病的根源分析，及其与药物副作用间的可能联系。他使用了一种叫做DAMI的算法，DAMI来自DataMiner（数据挖掘）的缩写。这是洛夫莱斯第一次接触遗传算法，程序在处理数据时，采用的是充满童趣的图形界面。洛夫莱斯回忆说："但其背后隐藏着令人难以置信的搜索算法，我从未见过这种处理数据的方法。"

雅各布森将这种算法的市场定位于广告公司和医学应用。洛夫莱斯建议他面向另一个市场，并与他在亚壁公司工作期间结识的一位朋友取得联系，后者那时候就职于华尔街的一家债券交易公司——康托・菲茨杰拉德公司。这家公司曾经的驻地是2001年"9・11事件"中遭受攻击的世贸双塔，事件中有太多的员工不幸遇难，公司也因此而广为人知。在人们的印象中，投资银行家总是衣冠楚楚、气度非凡。与此相反，华尔街交易所中，人们则要随意得多。每个交易员都面临同样的考验：要么为公司赚钱而收到奖赏，要么赔钱而被公司炒掉。外表真的不重要。

当洛夫莱斯与雅各布森来到康托公司后，公司给他们提供了大量美国国债的数据，并要求他们找出盈利模式。洛夫莱斯承认："关于美国国债，我们没什么概念，最多也就是小时候祖母给过我们的债券而已。出价、买入——我们根本不知道它们是什么。"为了填补知识上的空白，洛夫莱斯结识了康托・菲茨杰拉德的员工阿穆特・巴拉姆比。

巴拉姆比在印度中部的一座小城那格浦尔长大，从七岁起，他每天坚持六至九小时的专业网球训练，并周游全印度。但在十四岁那年，由于遭受致命的膝盖与背部损伤，他被迫提早结束了运动生涯。巴拉姆比将他的竞争优势带进了学校。他的父亲是一名统计学家，母亲是教育学博士，他们帮助他赶上了课程。2001年

他在印度读完了机械工程专业，然而凭借这样一个学位，他能做什么呢？他自己也不确定。巴拉姆比说："每个印度人都希望去美国和英国，凭借金融数学一类的背景，可以赚到很多钱。" 受到这一愿望的感召，巴拉姆比来到美国，并在纽约大学柯朗数学学院读完了硕士，专业是金融数学。一次实习的机会让他来到了康托·菲茨杰拉德的交易台边，仅仅比洛夫莱斯早来了几个月。

当时，康托·菲茨杰拉德公司经营美国国债交易业务，因此，原则上说没有任何风险，但变化即将发生。公司创建了"债务资本市场部"（DCM），这也是康托公司历史上第一个利用合伙人资本建立的部门，并承担相应的投资风险。在DCM中，有八位交易员和几个经理，巴拉姆比是DCM的咨询师。在此前的十年里，华尔街一直在做股票的程序化交易，但没有人把这种交易方式拓展到债券上。DCM为引入新的交易工具和交易手段提供了可能。巴拉姆比说："大卫·雅各布森和我一同挖掘数据，并找出所有这些有趣的模式。"

与此同时，洛夫莱斯则继续着他的咨询工作。他回忆说："在佛罗里达的海滩上，我配了一部电话，我就在海滩上接那些咨询电话。有一天下午，我正坐在冲浪板上，妻子打电话来问：'你还在海边吗？' 我说：'不，我在海里。' 她说：'你需要回去工作了。'" 于是洛夫莱斯打电话给康托·菲茨杰拉德的CEO，询问他是否可以做一名全职员工。此后，洛夫莱斯与巴拉姆比应公司要求建立了康托实验室，并扩展了程序化交易范围。面对海量的数据，洛夫莱斯决定使用K语言。在纽约，他真的一个熟人都没有，不过他发现了一个很受欢迎的K语言程序员社区。"这里有计算机科学中最优秀、最聪明的人。最大的对冲基金之一千禧年基金，其交易系统的整个后端由一名单枪匹马的K语言程序员建起来了。"洛夫莱斯认为他选对了工具。

巴拉姆比与洛夫莱斯开始建立一个完整的债券交易系统——涵盖了从市场价

格跟踪到发送订单的全部过程。系统还有一项重要功能等待开发，即找到有效的规则。无论是华尔街，还是别的什么地方，所有的数据挖掘类应用都采用相同的策略：将历史数据分成两组——训练组与测试组，然后用训练组数据找出规则，并用测试组数据对规则的有效性进行验证。数据分组是为了避免因偶然因素而产生无效的规则。虽然随机的一组数据也可以表现出某种规律，但必须有测试数据的参与，才能判定规则是否成立。

一个典型的规则也许是这样的："在连续两分钟内，如果十年国债的价格上涨了一定的量，而五年国债的价格保持不变，那么在接下来的两分钟内，五年国债的价格也很有可能上涨。"一个通过了测试数据验证的规则称为"通过回溯测试规则"。借助于统计学方法，也可以对规则的有效性做出评价，即，规则是偶然出现的（无效规则），还是可重复出现的（有效规则）。

因为是短线交易员，所以洛夫莱斯与巴拉姆比需要找到这样的规则：以前几分钟的行情变化为依据，预测出后5分钟或更短时间内的行情走向。具体要找到哪种类型的规则，他们也不确定。规则也许由一系列的属性组成，如每隔1分钟或5分钟，市场行情曲线的变化率；或者根据不同时间间隔内的移动平均数来构造规则。每个属性[①]都有多个值，一条交易规则就是这些值的集合。

由于时间序列[②]所描述的变量种类有很多种，而洛夫莱斯与巴拉姆比不确定哪些变量具有评估价值，于是他们决定从变化率[③]这个最简单的概念开始。最后，他们还要选择具体的变化率、行情数据字段及时间间隔，并将它们整合起来形成规则。洛夫莱斯回忆说："阿穆特只给了我28 000个属性值：连续两分钟内交易价格

---

① 将原始的行情数据进行某种数学或统计处理后，得出的代表行情趋势的次生数据，如变化率、移动平均数等。——译者注

② 在平面直角坐标系里，以时间为横轴，另一个变量（如股票价格、成交量、报价次数等）为纵轴所绘制的曲线。纵轴所表示的变量也称为行情数据字段。——译者注

③ 时间序列曲线的斜率，即一阶导数。——译者注

的变化率、连续30分钟内成交量的变化率，等等。他说：‘答案就藏在其中。’”

如果采用“暴力破解”法来寻找规则，为了得到所期望的结果，如历史收益的最大值，通常会在相关属性和数值的所有可能组合中进行探索。好的规则应该可以找到一些相应的条件，这些条件在两组不同的数据中表现出一致的结果，即，如果在训练组中导致了价格的上涨，就不会在测试组中导致价格的下降。但是那种包含了数千个属性的规则一般都是不可用的，由于它再现的机会极小或根本无法再现，因此也就无法进行统计测试。计算求解的目的在于找到这样的规则：使用相对较少的属性来获得尽可能多的预期结果。但问题是，即便是只有10个属性、每个属性有10个不同值的小集合，也会产生超过百亿个组合。为了解决这一难题，洛夫莱斯解释说：“我们需要考虑能够对数亿组合进行自动处理的办法，而不是花时间去考虑它们之间实际的关联关系。”

洛夫莱斯通过阅读大量资料来了解进化搜索系统的相关功能。系统开始时会随机选择一些属性和相关数值，从中生成一条交易规则，然后对照训练组数据进行回溯测试，并记录相应的收益或亏损值。接下来的一步有些难度：如何利用这些结果来指导下面的属性和数值选择？例如，假设收益规则1表述为：

如果1分钟的交易价格变化率>5%，且5分钟的交易规模变化率>10%，则买入。

而获利规则2表述为：

如果2分钟的交易价格变化率>5%，且10分钟的交易规模变化率<20%，则买入。

那么从这两条规则中能生成什么样的新规则呢？又如何对它们进行组合？你可能会随机地交换这两条规则的要素，来生成规则3：

如果2分钟的交易价格变化率>5%，且5分钟的交易规模变化率>10%，则买入。

或者也可以通过改变属性值来修改获利规则，如，将规则2中“2分钟的变化率”属性从5%改为10%，从而产生规则4：

如果 2 分钟的交易价格变化率>10%，且 10 分钟的交易规模变化率<20%，则买入。

洛夫莱斯使用了一些经典的规则搜索策略：如精英选择（classical elite selection）、局部平移（local shifting）、禁忌约束交叉（tabuconstrained crossover）等，以及在此基础上他自己发明的若干策略，并将这些策略与不同的适应度标准进行组合。针对每种组合方式都尝试构建一个解决方案的集合。当每一代的运算或每一次的迭代结束时，所有组合方式所产生的所有解决方案，都经过了评估及重组。

有些重组方法只是简单地根据得分的高低来交换属性及属性值，但洛夫莱斯显然想得到一个更加通用的系统。他解释说：“最终我们建立了一个通用的进化计算框架，可以并行处理大量的重组方法及不同的优化函数。”在每一代的计算结束时，所有搜索方法共享进化的结果，然后由一个第二级的启发式算法为下一代的搜索方法分配资源。

到我们写这本书时，经过多年的努力，洛夫莱斯与巴拉姆比已经积累了大约15种不同的重组方法和数十个不断扩充的优化函数，以及这些方法和函数所产生的有趣的结果。例如，他们发现，有这样一种搜索方法，它的优化函数用于寻找最低收益，从这种方法中获得的属性及相关数值，恰恰可以注入到其他寻找最高收益的搜索方法中。“进化计算将你带入一个奇异的世界。”这是洛夫莱斯给出的评价。

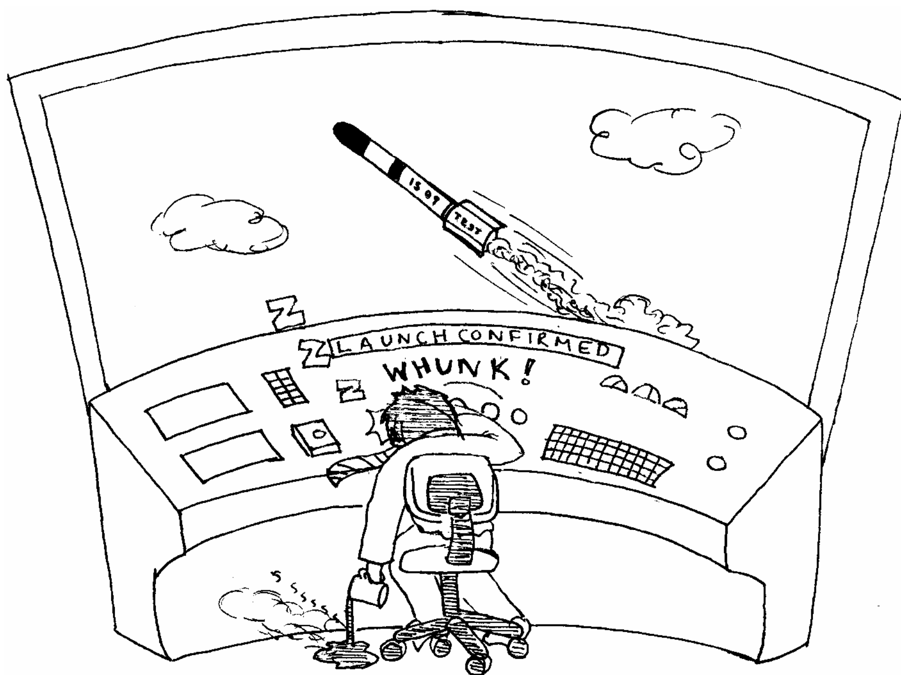
为了适应各种可能的情况，遗传算法必须找到数千条可能有效的规则。每个处理器每分钟对于每一代要测试超过10亿条规则。这种通用方法虽然不能保证给

出最佳的获利规则，却可以生成许多的规则。遗憾的是，洛夫莱斯与巴拉姆比对其中的多数规则都不能理解。“无论是10个、20个，还是30个，你都无法记住这些属性。不过我们最终还是实现了目标，这是一个真正的黑箱系统，我们不知道它下一秒钟是要买入，还是卖出。” 洛夫莱斯承认这一点。

最后，系统终于上线了。他们在某个周一的上午九点钟正式启动算法，并让它运行了一会儿，这期间系统买进、卖出了价值1千万美元的债券。洛夫莱斯说：“我当时一直在冒汗。在最初的两个月里，我每天都会带两件衬衫来上班。那是一种从飞机上跳下去的感觉，非常刺激。”还好，系统工作正常。

遗传算法对于处理类似的海量数据问题非常有效，即便是在2008年的金融危机过程中也是一样。但是身为冲浪高手的洛夫莱斯向往新一轮的浪峰。他接下来的目标是计算生物学，如探索蛋白质间相互作用的映射关系（更多内容请见本书第12章）。“那是唯一一个数据量超过华尔街的课题。”洛夫莱斯如是说。







你想让一个系统做什么，你不想让一个系统做什么，这是两个问题，是我们做事情的出发点，也是全部的安危所系。很多工程师考虑的是前者，因为他们接受过这样的教育；而我们要迫使他们去思考后一个问题。

——南希·莱韦森

第 5 章

南希·莱韦森

“这是系统，笨蛋”

1912年4月10日，“泰坦尼克号”邮轮驶离了英国的南安普顿港。报纸纷纷对这一船舶设计的杰作表示祝贺——它有三组螺旋桨、四座烟囱，有无线电通信、电灯以及一座游泳池。一些当时世界上最富有的实业家就在这艘船上，住在他们私密的、装饰奢华的头等舱里。但四天后，邮轮撞上冰山后沉没，造成重大人员伤亡。为什么一艘如此“完美”的巨轮会以如此不堪的结局收场?

事后调查表明，船只在夜间航行于北大西洋的冰山地带时，速度太快，而值班人员对冰山的瞭望却时断时续。除此之外，船上只为半数乘客配备了救生艇，更糟的是，许多本来可以搭乘40人的救生艇，仅装了12个人就下水了。对于这场海难的后果，有人将其归咎于船长和瞭望员，还有人认为是船的设计和行程安排没有考虑到星象的因素，恰逢一个不吉利的天时。有一个观点则认为，是过分的自负导致了船的沉没：他们盲目认为安全措施（充足的救生艇、严格的瞭望）对于一艘“永不沉没”的船来说是无足轻重的。

虽然技术一直在进步，但事故却总是与我们相伴。事实上，越是先进的技术，往往意味着具有更强的破坏潜能。通过对照因涡轮机叶片滑落，从而造成核电厂堆芯熔毁的后果，就可以得出这样的结论。来自麻省理工学院航空航天系的南

希·莱韦森分析了许多事故，尤其是那些夺命的事故。她相信寻找事故的直接原因，如粗心的船长及马虎的瞭望员，只会助长未来事故的发生。她坚持认为，对事故直接原因的过分关注，常常会掩盖掉引发事故的系统性原因。

莱韦森具有非比寻常的广泛的专业背景，这使她常常具有超越技术因素的视角。她认为，系统这一概念具有更大的外延——技术加管理、行业及企业文化、政府、社会及经济——所有这些她称之为“社会–技术”的因素，都会对安全造成影响。同时，她的研究涵盖了系统的整个生命周期，从概念的开发、设计、部署直至最终撤销。许多面面俱到的安全手段，最终都会被淹没在文字的海洋里，但莱韦森所倡导的方法论，为改进复杂系统的安全运行水平提供了可行性。

莱韦森出生于1944年，在洛杉矶西区长大。她的父亲是家里十个孩子当中最小的，也是唯一一个读过大学的。他在一家会计公司工作，之后便早早退休，呆在家里专心打理自己的投资。而莱韦森的母亲则经营一家自己的运动服装公司。莱韦森小时候很害羞，是个小书呆子，她自己学会了阅读（三岁时），并且喜欢翻看百科全书，那些书看起来比她的个头还要大。但是莱韦森的母亲想让她更多地与人接触。莱韦森回忆说：“我不得不藏起来，因为母亲总是催我出去，让我跟邻居家的小伙伴一起玩儿。”

凭借二年级时一次智力测试的出色成绩，莱韦森在学校跳了一级，这使得她比班里的所有孩子都要小：对于一个害羞的孩子来说，这种状况不是很理想。更糟的是，她开始感到厌倦。像许多聪明的孩子一样，面对平凡的老师、缺乏挑战的课程，莱韦森聪明的大脑开始溜号儿。她回忆说：“很多时候，我会一连几个小时做白日梦，创造出各式各样的世界。”在做完了高中的白日梦之后，莱韦森于1961年进入了加州大学洛杉矶分校（UCLA）。她主修数学，因为这个专业的必修完课程较少。这样，她就可以学习许多其他学科的课程，包括社会科学、人文科学、

艺术等。她把自己独特的研究方法归结于这种多学科的教育背景。

在UCLA，莱韦森开始逐渐融入人群。那是20世纪60年代中期，是一个政治和社会剧变的年代。学生们受到各种活动的吸引，这些活动使他们接触到了学术之外的世界。在UCLA的最后两年里，莱韦森大量逃课并且自学，只有在期末考试时才露面。

很多大学的数学专业会兼顾到理论及应用两个方面，但这两者之间却经常表现出冲突。UCLA的数学系以理论著称，但也开设了一门每周一小时的计算机语言课，使用的是一台老式的IBM 1620计算机。像对待其她课程一样，莱韦森报名选课，随后却忘了自己已经被选上。在期末考试的前一周，她收到通知，告诉她考试的时间和地点。她请求教授让她退掉这门课，教授深表同情，但拒绝了她的请求，并提示她考试是开卷的，她也有可能会通过考试。她考虑再三，认为没有必要花一周的时间来从头学习这门课，从而影响到其他课程的考试。她硬着头皮走进考场并开始阅读课本，这是她第一次阅读这门课的教材。最终她没能通过考试。莱韦森承认说："作为一名计算机专业的博士，却没能通过自己的第一门计算机课的考试，这也许是我的与众不同之处吧。"但花三个小时坐在考场阅读教科书的过程，居然让她产生了对计算机的兴趣，于是她决定将计算机科学作为自己今后的职业方向。她说："计算机就像我所喜欢的那种解谜游戏一样。"

接下来的问题是怎样说服研究生院，因为她的专业成绩实在是毫无说服力。那时的UCLA还没有设立计算机科学系，但在管理学院有一个计算机相关的项目，而且对专业背景要求不算苛刻。

她接触了一位做人工智能研究的教授，厄尔·亨特（Earl "Buzz" Hunt）。他同意接收她，但前提是她要上他的两门研究生的课——操作系统和人工智能，并取得良好的成绩。与周围的其他博士生不同的是，她几乎对计算机一无所知，而

且在整个八百人的学校里，她是唯一的女性。莱韦森承认说："我很害怕，不知道这两门课会学成什么样子。"这一次，她认真读书，上课积极提问，她回忆说："课堂上我几乎一直在举手，班里的每一个人都认定我会不及格，包括我自己，也包括教授。这个'女孩'来这里完全是个错误。"然而，莱韦森证明所有人都错了，她获得了这门课的最高分，亨特同意做她的博士导师。但周围几乎没有女性，这让莱韦森感觉到不自在。她也从未见过女教授，似乎女性不适合留在学术圈里。

莱韦森于1967年完成了自己的硕士学位，她决定去洛杉矶的IBM公司当一名系统工程师。那时的IBM向其客户提供一站式购买服务，即同时提供硬件、软件开发，甚至还包括操作方面的技术支持。莱韦森的工作除了以上各项外，还包括调试操作系统、指导其他人使用新的信息工具及数据库系统。莱韦森很享受这份工作，收入也很丰厚，但逐渐地她又不安分起来。于是她从IBM辞职，决定用赚来的钱去环游世界。

两年时间里，莱韦森一路搭便车绕着地球转了一圈半，平均每天的花费是50美分。她曾经与一对澳大利亚夫妇结伴而行，并住在新几内亚高地的一个石器时代的原始部落里。此时，她开始审视自己的生活。她把自己在亚洲的见闻与在美国的生活进行比较，莱韦森说："在西方，我们感觉世界围绕我们每一个作为个体的人而旋转，我们把自己的生命看得比什么都重要。当你来到亚洲，看到大量居民生活在极其恶劣的环境里，你开始感觉到，在这天地之间，自我是多么渺小，多么无关紧要。"她意识到自己应该从现在开始，做一些改变自己、对人生有意义的事情。

莱韦森回到美国，开始在高中教数学课。一年后，她决定去教那些心理有障碍的孩子。她参加了UCLA的一个发展心理学项目。尽管她在学术方面做得很好，但她发现自己并不擅长与孩子相处。另一个担心是，她要花八年时间来取得一个

认知心理学的学位，而这个领域的工作机会非常少。朋友们建议她回到计算机专业，用三年时间去拿到一个博士学位。于是她转回去学计算机科学的人工智能专业，却发现自己已经不再喜欢它了。她觉得在20世纪70年代，人工智能的研究者们所遭遇的瓶颈，与20世纪60年代并没有什么差别。

莱韦森再次转变专业，这次她选择了软件工程及编程语言专业。她在1980年完成了学业，并在加州大学埃尔文分校得到了一份工作，她是系里的第一位女性。"想成为'第一'并不容易"，莱韦森说。她仍然不能发表自己的学位论文，因为论文遭到了剽窃，剽窃者此前曾经向她索要过这篇论文的早期版本。莱韦森回忆道："我非常沮丧。"在埃尔文分校工作的第一周，莱韦森接到了休斯飞机制造公司的一位工程师打来的电话，他的工作是用15个微处理器来控制鱼雷。他们有些担忧——不是因为无法击中敌人，而是因为鱼雷有可能旋转180度回来击中发射它的船只，这种误伤是一种极大的侮辱。莱韦森告诉他，自己研究的是编程语言的形式语义，而不是可靠性："我对鱼雷一窍不通，我是搞应用数学的。"她回忆道："他说，'我们已经获得了经费。'我说，'那我可以学。'"于是莱韦森又开始接受与系统安全有关的教育。

在商用航空及核电行业，安全工程的目标是建立系统的"失效保险"（fail-safe）机制。大概意思是，在确保每个部件都非常可靠的前提下，要保留若干个完全相同（冗余）的备件来增强可靠性。商用航空及核电领域的工程师们同时也强调要从以往的意外事故中学习——称作"飞-停-飞"方法。这种方法之所以有效，是因为系统的设计相对简单，而且较少变动。直到现在，除非特别需要，这些领域对计算机的使用仍然是十分谨慎的。

安全问题在国防工业中则极为不同。武器总是伴随着最先进的技术而不断更新。从20世纪50年代的早期洲际弹道导弹（ICBM）系统开始，计算机就被用来控

制那些非常复杂的系统。尽管有良好的备件和充分的冗余，但意外事故和未遂事故仍然时有发生。工程师们发现，复杂系统中的多数安全问题，均产生于组件之间的协作过程，而非独立组件的单点故障。因此仅有冗余是不够的，它无法避免软件错误所引起的故障。一个设计上不正确的软件，给它再多的版本更新，也不可能产生正确的行为结果。

为了解决复杂的国防系统中的安全问题，一种新的安全工程模式应运而生，即“系统安全”。系统安全的核心思想是：在系统的建设之初就将安全纳入到设计之中，而不是在系统建设完成之后再扩展其安全功能。系统安全关注的是系统的危险状态，或非安全状态，比如一次意外的导弹发射。借助于灾害分析来推测此类危险状态的形成过程，从本质上讲是一种事前研究的方法。科学家依据这些研究结果对设计进行调整，从而消除灾害隐患。

莱韦森多元化的专业背景，包括社会科学背景，让她如虎添翼。她有能力将事故相关的社会及文化因素与工程技术进行整合；而她的计算机科学的背景则有助于她解决复杂的系统工程问题。她说：“数字革命在给工程师们带来惊喜的同时，也动摇了他们脚下的基础。”他们的许多基本假设（如不同组件的故障是互不相干的）不再适用于计算机及软件，在这里，一直都存在着相互关联的系统故障。

像大多数的安全分析师一样，莱韦森曾经尝试将传统的安全工程技术延伸到软件领域。她考虑过对软件进行安全测试，却发现这个方法行不通。莱韦森解释说：“事故的发生通常是因为你遗忘了某些关键的事情，或者是因为对系统及其运行环境的假设根本就是错误的。”而测试过程本身就会受到这种错误假设的困扰，因此总是显得过于滞后。对于复杂的软件系统，试图在测试阶段做重大改动，将极大地延误工期并加大成本。

于是莱韦森将着眼点从软件测试转移到软件设计，然而她很快就意识到，大

多数与软件有关的事故是由需求本身的错误所造成的，而非由于软件的设计。得出这样的结论其实一点儿都不奇怪："泰坦尼克号"的需求就存在错误：如果设计师在设计时考虑了救生艇，那么为什么不能为全体乘客及船员预备充足的救生艇呢？于是莱韦森开始解决需求领域的问题，并从改进人机交互开始着手，因为软件的操作者经常会抱怨说，是软件的操作让他们感到困惑，或软件的提示误导了他们，才导致事故的发生。莱韦森解释说："一旦接触到了人为因素并进入需求层面，我马上意识到自己应该抛开计算机，回到系统工程的角度去思考问题。"

计算机的出现，使得人们得以建造更加庞大而复杂的系统，同时也给系统工程带来了许多新的问题。莱韦森说："我们无法再借助于物理定律来限制这些设计的复杂度。"在普遍使用软件控制之前，工程师们采用物理的制约方式来确保安全。例如，一辆传统的火车虽然采用了电控刹车，但在设计上依然保留了机械的轮制动功能，以便当电控失效时，火车依然可以借助重力的作用确保安全。然而对于像软件制导鱼雷、空中交通控制或金融管理这样的系统，没有哪一种自然力可以确保它们的安全。对于无所不能的软件来说，"失效保险"机制的设计理念显得无能为力。

伴随着她"不断搜寻更加恶劣天气"的步伐，莱韦森先是离开埃尔文分校前往西雅图的华盛顿大学，随后又来到MIT的航空航天系。她爱上了MIT，因为在那里她平生第一次发现了这样一群人，他们能够在更广阔的背景下去思考系统问题。莱韦森现在正在研究大型系统中她称为"社会–技术"方面的问题，这一问题将工程学与管理及社会科学整合在一起。她的最新著作名为*System Safety Engineering: Back to the Future*。书中她追溯了早期系统安全先驱们的思想，并希望将它们应用到系统工程的前沿，以期找到新的出路。

莱韦森面临的是一场艰苦的斗争。她说："工程学专业的学生受到的教育强调

可靠性，而不是安全。”由于在系统安全领域缺乏正规的教育，因此工程师们不得不在工作中学习，向其他系统的安全工程师们学习。可靠性与安全听起来很相似，但其实存在着极大的差别。例如，如果检查一个核电厂的阀门，你很容易判断该阀门是否可靠，但你无法预测整个核电厂是否安全。莱韦森说：“整体的安全依赖于其组成要素之间的相互作用。在系统理论中，安全被理解为某种涌现（emergent）[①]的特性。”

人们通常会以蚁群为例来说明“涌现”这一概念。虽然经常有个别蚂蚁死掉，但整个蚁群却能得以幸存。蚁群成员都知道如何担当那些特殊的职责，如搜集并储备食物。同样，在计算机系统中，假如我们有足够数量的计算机，并且它们之间存在适当的交互，那么即使使用了不可靠的组件，依然有可能实现安全性。相比之下，对于一个没有涌现机制的系统，即使是一个很小的失误，也可能会导致严重的事故。

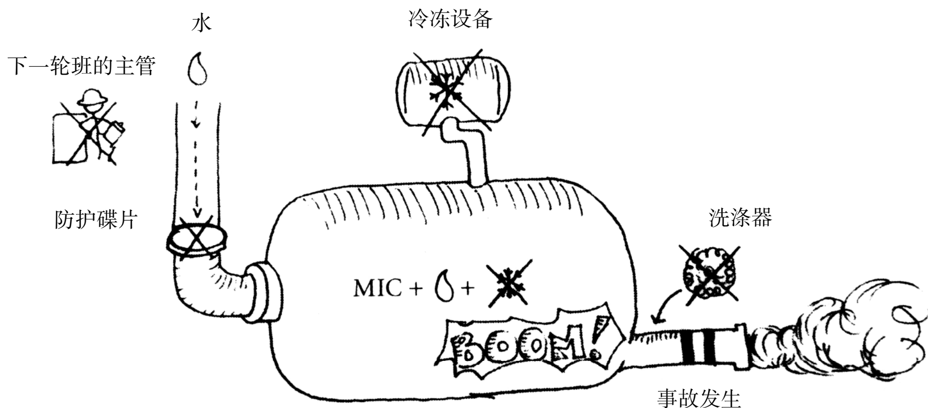
工程师们习惯于将事故的发生归咎于某个“根本原因”，它存在于引发事故的一系列事件之中。莱韦森则认为这种方法既过分简单化，又贻害无穷。因为最便于识别的根本原因通常位于事件链的起点，而且通常也只会认定一个根本原因；与此同时，那些真正促成事故发生的系统化因素却常常被忽略。莱韦森以博帕尔化学品泄露事故为例，来说明系统性的因素如何引发一系列不可抗的事件。

1984年12月，异氰酸甲酯（MIC）[②]从印度博帕尔市的美国联合碳化物公司的化工厂释放出来，造成至少1万人死亡，超过20万人受伤，使其成为历史上最惨重的工业灾难。事故的直接原因是将清洗管道和过滤器的工作分派给了一名新工人。在清洗过程中，这名工人由于疏忽，忘了插入一个专用的碟片，来防止水渗入MIC

① 涌现一词用于描述复杂系统的一种重要特性，即，构成复杂系统的次一级单元之间通过简单的相互作用而呈现出复杂的结果。——译者注

② 易燃、易爆且具挥发性的剧毒液体。——译者注

储藏罐。结果水流进了储藏罐并导致了罐内的压力激增，然后致使一个减压阀打开，40吨的MIC被释放到了人口聚居区的上空。



在博帕尔市的化工厂事故中，大多数设计好的安全机制（下一轮班的主管、冷冻设备、洗涤器）全都失效了

究竟什么才是这次灾难的真正原因？是一个没有经验的养护工人的疏忽大意吗？或者，像联合碳化物公司化工厂之后所声称的，是某个蓄意破坏活动的一部分吗？莱韦森相信这些解释无一切中要害。管道清洗工作应该是在下一轮班的主管监督下进行的，但这个职位却因为要压缩成本而被取消了。而且插入碟片的工作也不是管道清洗工的职责，因为他是个职位较低的雇员，不应该担当如此重要的任务。退一步说，MIC被水污染的确很糟糕，但是如果能够像当初设计所要求的那样，将MIC冷藏，那么也不至于引起如此剧烈的压力激增，然而冷冻设备却一直都没有运行。而且MIC这种物质本身也不应该从管道泄漏出来，因为洗涤器可以阻挡它，但不幸的是洗涤器也停运了。

在莱韦森的分析中，养护工人在整个事故中只是一个微不足道的角色；相反，真正的原因在于长期积累所造成的安全状况恶化，以及对这些状况的熟视无睹。一系列的决策将工厂逐渐推向深渊的边缘，以至于任何轻微的失误都可能导致重

大的事故。尽管印度政府把事故归咎于人为失误，但莱韦森指出，是整个系统的不安全才导致了事故的发生。失误在所难免，问题是系统应该如何来设计和运行，才能避免一般性的失误或失败酿成重大的灾难性事故。

将事故原因归咎于操作者的失误，极大地解脱了公司及印度政府的责任，而印度政府拥有该工厂26%的股份。同样，把事故原因归咎于一次蓄意破坏活动，则全盘解脱了管理者的责任。相反，莱韦森的观点强调，必须对整个工厂的运营体系进行重大改革。只要工厂处于运行状态，管理者就必须确保最初设计的灾害防控系统（制冷装置、洗涤器等）同时处于正常的运行状态。这次失败是管理者的失败。

系统理论把安全视为整体控制的问题，而不是针对具体的故障。一方面，通过对系统及其组成要素的行为施行强制性约束来控制整体的安全性，另一方面，控制的实现也要依赖于反馈机制。想象一下，在旅馆里遇到了一个样式奇特的淋浴器，你会怎么做？你会把调温器转到一个适当的刻度，感受一下水温；如果不合适，再旋转调温器，再试试水温；如此往复，直到合适为止。即使是再奇特的调温器，你也能很快把淋浴器调到一个舒适的温度。基于反馈的控制的唯一也是最重要的关键点在于传感器（如你的手）必须是精确的，同时控制功能只能是“单调”的——当调节器沿顺时针方向旋转时，总是产生同样的结果：水温降低。

有效的控制需要准确地掌握过程中存在的各种状态，同时具有实施控制的能力。博帕尔化工厂对管道清洗的操作过程缺乏足够的控制——那个被取消了的督查员本来应该履行他的安全检查职责。而由于洗涤器的年久失修，工厂没能对已经泄漏的MIC实施任何控制。同时，对于可能存在的危险，附近的居民从未接受过警告，更没有接受过最基本的逃生技能的指导，来阻止伤害的发生（如用湿毛巾捂住脸等）。这些还只是问题中的极少一部分，要真正了解事故发生的原因，还

需要透过工厂，去分析联合碳化物公司及印度政府对管理的疏忽。全方位地审视事故的原因，有助于防止类似事故的再度发生。

在对事故的研究中，莱韦森发现了一些共性。首先，通常都存在安全操作等级降低的问题。如，由于许久没有出问题而使安全人员变得盲目乐观、管理上的削减预算以及设备的年久失修等。博帕尔灾难提供了许多这方面的例证，但这种情况并非印度所独有。例如在南加州，暴雨经常导致洪水泛滥，由于某些地区的雨量稀少，因此城镇的居民忽略了排水沟的疏通工作，于是几乎每逢下雨就会导致洪水泛滥。

第二，事故的发生还常常是源于改进工作的不同步。设想某人升级了软件（出于私人目的而进行的优化），然后发给你一份文档，文档是用新版文字处理软件编写的，与你当前的文字处理软件版本不兼容，你无法阅读这个文档，因此交流出现了障碍。有时候这种局部的升级优化，甚至可能导致无谓的生命及财产损失。例如，在1994年，两架美国战斗机击落了一架美国直升机。由于战斗机使用了更新的通信系统，而直升机还在使用旧系统，因此直升机飞行员未曾收到过来自战斗机的警告呼叫。

第三，系统组成要素之间的协作失调与交互障碍也会导致事故的发生。单独来看，各个组成要素的操作都正确无误，但在总体上它们的行为却惹出了麻烦。例如几年前在德国南部，两架飞机在半空中相撞。瑞士的空中交通指挥员向飞行员发出了正确的指令，使两架飞机可以相互避开；然而与此同时，机载的防撞系统也向飞行员们发出了指令。如果两名飞行员都听从同一套系统的指令，结果应该是相安无事。然而，一位飞行员听从了机场的空中交通指挥员的指令，而另一位却听从了机载自动防撞系统的指令。

这三个问题——安全操作等级的降低、技术更新的不同步以及协作与交互的

失调——明显是相互关联的。大部分这类问题都发生在一套常规的流程被建立起来之后：系统中的个别要素出于寻求更高效率的目的，或私自削减资金预算（博帕尔），或对自有的运行系统进行优化（战斗机）。这样做的结果是弱化了控制，并导致了事故，就好像在调节淋浴器温度的同时，不用手去测试水温一样。

莱韦森的安全驱动设计思想来自于对这些问题的观察。她主张，在建立系统之初，就要采用她所制定的基本系统工程原则和灾害分析技术，将安全融入到整个系统的设计之中。在最近的著作中，莱韦森提供了一个简单的案例：火车自动门系统。设计过程的第一步是确定系统风险。以下罗列出部分风险：(1) 某人被关闭中的车门撞到；(2) 有人从运行的火车上坠落；(3) 乘客和乘务人员无法从车厢内的危险环境中逃离，如火灾。这些风险因素可被转化为以下约束条件，用于自动门及其控制系统的设计：(1) 被阻挡的车门必须重新打开，并允许阻挡物移开，然后再重新自动关闭；(2) 车门必须在火车停稳之后才能打开；(3) 无论火车停在何处，都必须为紧急疏散提供必要的开门设施。

从这些约束条件可以导出各种不同的设计方案。例如，第一个约束可能会提示设计者在车门上使用类似于电梯门上使用的那种传感器；第二种约束可能需要在运动传感器与车门传感器之间建立某种联锁机制；第三个约束可能需要在警报响起时，将联锁机制与“开门”指令结合起来。事实上，是由风险因素生成了约束条件，又由约束条件生成了设计依据，目的是从一开始就为系统植入安全性。

让安全问题在设计的各个层面上都能获得充分的理解，从最抽象到最具体，这样就可以杜绝经营管理中以降低安全为代价的成本优化行为。“反馈并控制”的方法远远好于“假设故障不存在”。总而言之，如俗话所说，不怕一万，就怕万一。

莱韦森已经将系统安全理论应用到了美国导弹防御局，那里构建了一套巨型的系统间的互联系统。其中的有些导弹防御系统可以追溯到20世纪60年代，包括

冷战时期的北美航空航天防御司令部（NORAD）。其他一些则是近现代的，如新式雷达机平台。莱韦森应邀参与评估工作，不仅要评估每个部分的可靠性，更要对整体的安全性做出评估。“分析发现，在新的国家导弹防御系统中，有太多的因素会导致意外发射，为了修复这些错误，系统的测试及发布时间被延长了6个月。”

莱韦森的系统方法同样也应用于各种民用领域的问题，包括药物的安全问题。例如，药物安全从根本上讲依赖于化学及人体生理学，但发现问题却依赖于适当的检验。药物安全的相关因素包括预期的目标人群、测试人群的人口统计学指标及总体健康水平，以及相关的质量控制等。然而，检验过程本身的质量则要依赖于监管机构的经费及监管力度。监管力度会受到医药公司的游说力度的影响。说客所在的医药公司利润越丰厚，说客们能花的钱就越多。因此，当药品行业的利润日渐丰厚时，说客们的出手就会更大方，立法者们就更容易降低监管要求，检验也变得更加宽松，而药品就会更加快速地流入市场，因此医药公司利润的增加与立法监管的放松就更加变本加厉。在某一时刻，某种缺乏检验的药物会引起明显的伤害，从而招致强烈的抗议并最终导致过度监管。对莱韦森来说，问题不在于劣质的药物，而是在于这样一个事实：检测质量和游说国会处于同一个控制回路之中。检测质量应该只取决于药物对公共健康的影响因素，而非药厂的游说力度。

正负反馈控制回路的例子在自然界中随处可见，正是它们使生物保持健康，并使生态保持平衡。系统安全尝试在人造系统中也能够实现这一机制。一个安全先行的设计可以通过结合反馈控制回路来阻止灾害的发生，同时也将系统安全引向了自然计算。



第二部分

驾驭生命物质

如果有人提到“计算机”这个词，你会想起你的笔记本电脑、手机，或许还有你车里的数以百计的嵌入式微型计算机。那么如果要你来解释它们的运行机制，你可能会想，是电力提供了能源，晶体管构成了开关元件。不错，这是当前计算机的主流运作模式，但计算和电子技术之间其实没有必然的联系，毕竟，是人在进行计算。

我们人类如此地善于计算，这或许是因为那些构成人体的最基本的生物单元——DNA、病毒及细胞都能做计算。想象一下，如果我们可以控制这些构造单元，并能用它们来计算，那会怎么样？那么，到底有没有可能呢？

内德·西曼是纽约大学的化学教授，他在历尽挫折与无奈之后，最终奠定了DNA纳米技术的基础。他之前一直尝试蛋白质的结晶，这是一项非常困难而且枯燥乏味的工作。有一天，一位同事请他帮忙分析一种DNA霍利迪交叉，这是一种自然形成的不稳定的DNA，具有四路交叉结构。在了解了这种连接方式后，西曼意识到，他可以用DNA创造出任意形状的结构，而不仅仅是主流观点所认为的线性双螺旋结构。从此，他开始在分子水平上建造这些结构，包括DNA砖块和10纳米级的机器人。

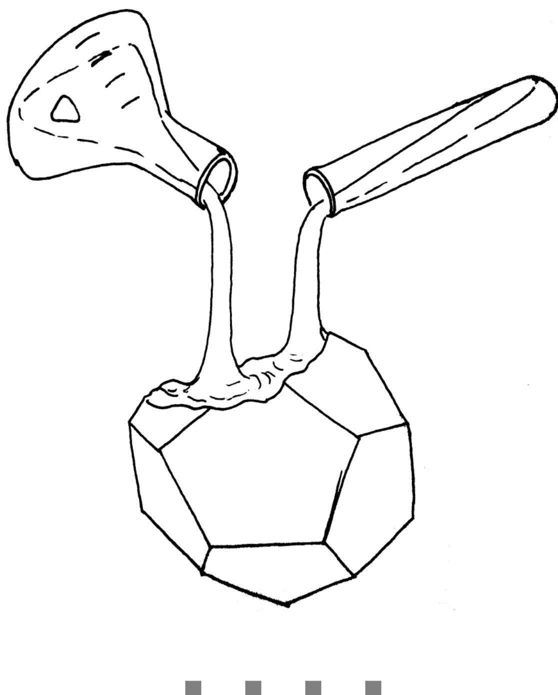
西曼的技术基于这样的事实：两条DNA单链可以相互结合，只要组成两条单链的DNA字母表在排列上能够相互对应——一支链上的A与另一只链上的T相对；同样，一支链上的C与另一只链上的G相对。这一技术的直接效果就是：他将几小瓶的DNA单链混合在一起，利用一种叫做“自组装”的强大技术建造出了一种结构。设想在没有人力介入的情况下，能够构建出一个非常精确的几何形状，这就是自组装。

西曼的这一重大发现引发了与动态DNA相关的一波研究浪潮。加州理工学院的年轻研究员保罗·罗特蒙德，演示了如何将一个完整病毒中的有7000个碱基的DNA折叠成一个笑脸的形状，这同样是利用了自组装技术。演示证明，即使是大尺寸的形状也可以用自组装的方式来创建，这一点为高精度电子线路的微加工奠定了良好的基础。罗特蒙德的努力除了具有重大的经济价值，更激动人心的是，他可以将特型DNA的大型片段植入到细胞中，这将产生一种全新的研究方式，甚至可以改进细胞生物学，以服务于医疗事业。

当罗特蒙德以病毒作为DNA自组装的原料时，史蒂夫·斯凯纳和他在纽约州立大学石溪分校的生物化学同事们修改了一个真正的骨髓灰质炎病毒，希望以此来减弱病毒的毒性，从而产生出疫苗。通过给受试者注射少量疫苗，在不伤害到感染者的前提下，疫苗激起了感染者的免疫反应。身体会“记住”这次反应，当今后再次感染同一种病毒的更强的变种时，免疫系统会识别并消灭这一病毒。筛选出弱的病毒是一件很困难的事情，要经过不断的试验和失败，就像20世纪50年代约纳斯·沙克通过对猴子进行试验而发现骨髓灰质炎疫苗时一样。斯凯纳用病毒自身的DNA编码来设计病毒，病毒的繁殖速度非常缓慢，这使得它的毒性得到减弱，但依然可以引起适度的免疫反应。斯凯纳的技术利用了遗传密码的非对称性，因此可以降低病毒蛋白质的合成速度。

DNA可以自组装，病毒可以嵌入到细胞里，但是细菌和多细胞生物拥有更高的能力：它们可以繁殖和自组织。这些能力提供了一种可能性，即利用数百万的细胞，在细菌水平上完成大量的计算。一般我们把多个处理器一同工作叫做并行处理，但是这里所说的细菌水平上的计算，采用了数百万甚至数十亿个计算单元，其并行处理能力超越了以往任何基于硅结构的处理系统。杰拉尔德·萨斯曼认为，既然细胞可以共同发育成胚胎，那么也一定可以被组织起来对自然进行改良。他设想用细菌来构造没有缺陷的木材、调节润滑油的粘稠度，或者修复我们的身体损伤。

萨斯曼的一个从前的学生，拉蒂卡·纳格帕，已经实现了一种面向细胞类对象进行编程的方法，其挑战在于要驱使这些脆弱的、异步的、只能与最近的邻居进行对话的细胞，去共同达成某些全局性的目标。纳格帕的方法借用了化学扩散的思路，即在一组排布密集的同类细胞中，如果从某一点发出了一个化学信号，那么该信号将以几乎相同的速度向所有方向扩散。化学信号扩散的这个特点提供了一种方法，可以实现数百万脆弱的生物细胞、微型机械设备或者微小机器人的协同工作。指令通过细胞培养液以近乎一致的速度向各个方向传播，虽然会有个别细胞自身速度与别的细胞有差异，甚至有些细胞可能会死掉，但不会影响到整体的效果。全局性的指令致使数百万个细胞协调一致地折叠成精确的形状，并保持平衡，甚至还可以造出某些新玩意儿。



■ ■ ■ ■

我一直都把DNA序列当成是一个长长的由四个字母组成的单词。它是个一维的对象，而我却不是个一维的家伙。

——内德·西曼

| 第 6 章 |

内德 · 西曼
生命的边际

内德 · 西曼一开始是研究晶体学的：通过X射线的衍射实验来了解底层的分子结构及其排列规律。经过人们多年的努力，晶体学研究已经是我们了解材料科学及生物学最重要的手段。在19世纪80年代，皮埃尔 · 居里和他的哥哥雅克证明了晶体的变形会导致电位的升高，升高的程度取决于晶体的类型和质量。这项发现形成了现代半导体业的基础，只是现在这个行业对晶体要求更加完美。在生物学中，晶体学家的实验数据使我们认识了生命的基本组成单元。在20世纪50年代，罗莎琳德 · 富兰克林针对DNA的X射线衍射数据帮助沃森和克里克发现了双螺旋结构。

西曼后来不再是一个纯粹的晶体学家，他和他在纽约大学的学生花费了大量时间来设计化学实验规程，用于诱导DNA形成分子雕塑及机器人。未来，这些雕塑可以形成一个支架，用于建造纳米电子设备。或者，它们也可以变身为微型机器人，来创建纳米量级的工厂。目前，全球大约35个实验室都在从事DNA纳米技术的研究。西曼早在20世纪80年代就开始了这项技术的全面研究，其中很大一部分原因是出于他对所谓“单调乏味且愚蠢的晶体实验”的强烈抵触。

西曼于1945年出生在芝加哥，母亲根据西曼爷爷的名字南森为他取名为纳德

里安（Nadrian）。西曼说："爷爷讨厌他自己的名字，而母亲也为我取了一个让我自己厌烦的名字。"西曼在正式出版物上使用纳德里安这个名字，但他更喜欢人们叫他内德（Ned）。在芝加哥郊区的高地公园读高中期间，西曼的生物老师用物理及化学现象来解释生物问题，他为此深受启发，其中遗传学的那部分吸引了他的注意。他回忆说："从那时起，我的兴趣就定位于生命与非生命系统之间的天然分界线，即生命的边际。"西曼的父亲是个皮货商人，他想让自己的儿子成为一名医生。于是在西曼按照这个计划进入芝加哥大学读书。但在见证了祖母病危时饱受磨难而医学界却束手无策之后，年轻的西曼决定不再学医。

于是他转去学习生物化学。在20世纪60年代中期的芝加哥大学，这就意味着要去测量小分子的代谢反应。西曼回忆说："令人难以置信地无聊！！"他于1966年来到匹兹堡大学，这次他决定首选那些更偏重数学的学科，并进入了晶体学专业。西曼说："在晶体学专业，可谓一帆风顺。这里的课程有计算、三维晶体学，还有一些数学和物理方面的课程，这些对我来说都易于理解，而且后来发现我居然很擅长这些科目。"

西曼在1970年获得了晶体学/生物化学博士学位。他在哥伦比亚大学做博士后期间，（为一个RNA分子）解析出了他的首例核酸结构（参见附注栏内容"如何解析晶体结构"）。之后西曼在MIT的阿莱士·里奇实验室工作，继续从事核酸组分的晶体学研究。"这是一种硬质晶体结构，对我来说是个理想的课题。"西曼回忆说。

如何解析晶体结构

要理解西曼在解析RNA分子结构上的成就，就必须懂得解析晶体结构的基本原理。X射线晶体学基于这样的事实：穿过晶体的X射线会受到晶体中电子云的衍射（向各个不同方向的反射），由衍射条纹可以获得电子密度的信息。X射

线数据也给出了复数傅里叶分量中的振幅信息，也就是说，这些数据用数学方法描述了晶体结构的重复样式。但数据中不包含相位信息（重复样式的偏移量）。“解析晶体结构”意味着要找出相位。结合振幅与相位，你可以重建晶体结构。也就是说，可以画出晶体中所有原子的位置。

当西曼开始工作时，学界存在这样一种观点：如果晶体由许多电子数大致相等的原子组成，那么分析晶体的结构是非常困难的。西曼要结晶的序列（腺苷酰磷酸尿苷：ApU）中包含了95个（包括溶剂）非氢原子（氢在X射线中几乎是不可见的）。其中的两个非氢原子是磷（15个电子），其余的都是电子数少于或等于10的第一行原子[①]。因此这是一个很难分析的结构。术语“第一行”（first-row）指的是元素周期表的位置。如果有一个或两个非常重的原子，如碘（有53个电子），那么很容易确定它的位置，因为一个原子的信号强度与原子核外电子数的平方成正比。从这个原子出发，就很容易确定结构的其余部分。

西曼结晶的序列为腺苷酰磷酸尿苷。在这个非对称单元里有两个分子，而每个分子中的每个原子的坐标必须被确定。这个序列包括一条位于两个糖基单体之间的磷酸主干，因此系统中有两个A-U碱基对。在当时，这是晶体学研究中见过的最大单元，而此前尝试过大约20个类似的分子都失败了。

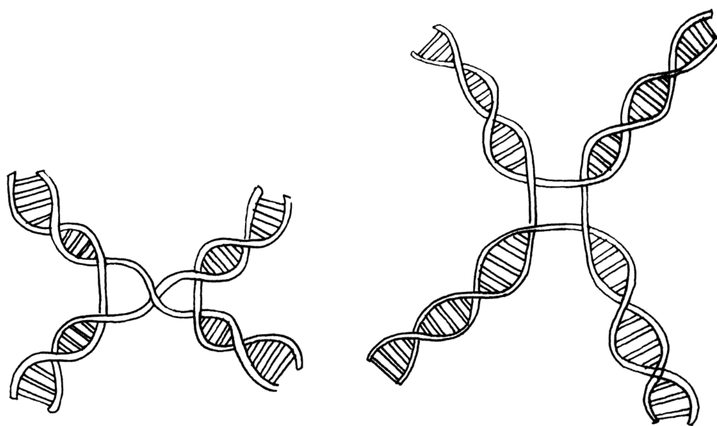
6

当西曼申请成为一名晶体学的助理教授时，就业市场非常不景气。晶体学的研究成本昂贵，没有一个学校会想再要第二个晶体学家。在他那年，只有百分之六的晶体学家能找到工作，而西曼无疑是找到了一份好工作。“但是在纽约州立大学奥尔巴尼分校工作，就像是被判了一次科学的死刑，在那儿，我度过了3983天，我常常将它四舍五入为4000天。”他说。

① 电子数少于或等于10的原子分布在元素周期表中的第一、二行，而不仅仅是第一行。此外，第一行的非氢原子只有氦，而ApU中不含有氦原子。因此原著中的“第一行”应该为“第二行”。
——译者注

在那些日子里，研究晶体学的每个人都会选定一种蛋白质，然后年复一年地研究它，当时的技术条件也只能这么做。西曼对于蛋白质的兴趣并不像对DNA和RNA这些核酸那样浓厚。尽管核酸中包含了一些生成蛋白质的逻辑，但西曼还是无法施展他的晶体学魔法：生长出一个大型的晶体。西曼回忆说："看来我得像纽约城里那些出租车司机一样，给困在那里不停地打转了。"

然而他终于时来运转了。有一天，与西曼同在一层楼办公的博士后布鲁斯·鲁宾逊来找西曼，转达了他的老板伦纳德·莱尔曼的求助，莱尔曼想请西曼去看一种叫"霍利迪交叉"的DNA分子结构。（莱尔曼以发现了"DNA嵌入"而著名，即一种将非DNA分子封装进DNA分子的工艺。）霍利迪交叉是一种类似于四车道交叉路口的四链螺旋结构。在自然界，稳定的DNA具有线性的拓扑结构。实际上，莱尔曼与鲁宾逊是想请西曼帮忙建立一个模型，来解释非线性的霍利迪交叉在什么情况下出现，又如何会消失。合作进展得很顺利，从观察得知，由于结构的对称性，霍利迪交叉可以从一种构型转变为另一种构型，但西曼的兴趣却远不止于此（见插图）。



四路DNA交叉的两种可能的构型。第一种情况：右上与左下的螺旋相联接；第二种情况：右上与左上相联接。两种情况下的沃森–克里克配对引力是相似的，因此两种状态都可能发生

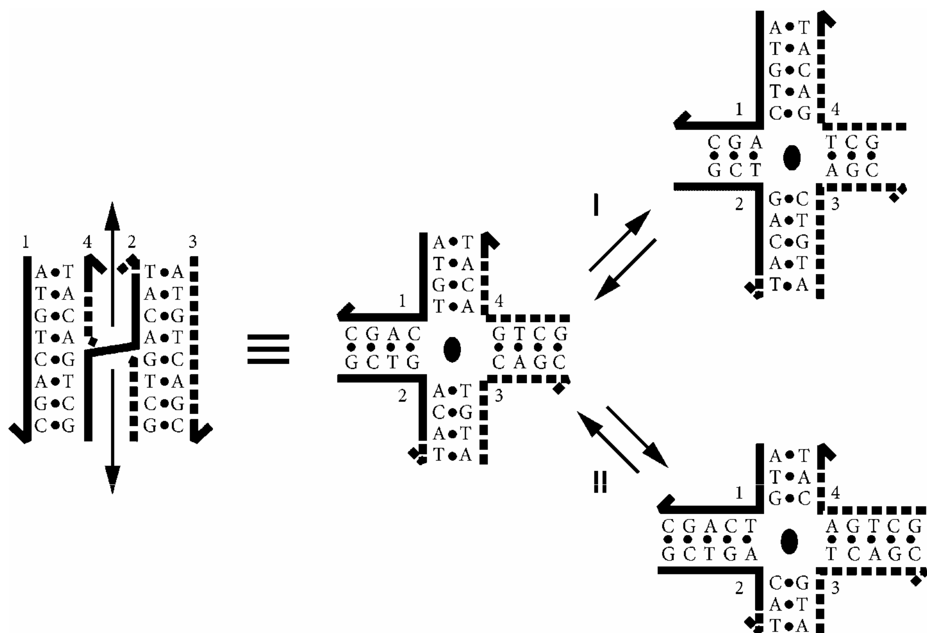
西曼在与他的本科学生凯西 · 麦克多诺交谈时意识到，他的团队可以针对上述模型所产生的某些假设进行验证。他推断，如果他们可以人工合成DNA，那么就可以摆脱霍利迪交叉的对称性，并因此创建一个稳定的联结。一个稳定的、非线性的DNA结构之前还从未出现过，而这正是西曼在DNA纳米技术领域研究的开始。

西曼意识到四臂分支结构仅仅是个开始，如果能够制造出一个稳定的分支结构，他们就能增加臂的数量。1980年9月，西曼来到校园酒吧思考六臂联结问题，荷兰艺术家埃舍尔的画作《深度》启发了他。埃舍尔的画作表现了一个多维度的鱼的学校，每条鱼都有六个船桨一样的鳍。西曼回忆说："我意识到这种飞鱼刚好就是六臂联结的样子。重要的还不仅是埃舍尔画中的鱼具有六臂联接的拓扑结构，而是这些鱼的排列也如同分子在分子晶体中的排列一样。它们在三个维度上的排列都具有周期性：前后、上下、左右。"他认为他可以采用这个思路将DNA组织成晶体。

合成DNA在1980年是一件苦差事。西曼解释说："你有一种分子，但你不知道靠什么样的相互作用才能把晶体结合在一起；就算偶然有一次成功了，你也不知道为什么会以这种方式结晶，而不是其他的方式。"但他认定自己的目标——理性结晶是值得去努力的。理性设计DNA结晶的基础是"沃森–克里克配对法则"：核苷酸A（腺嘌呤）要与T（胸腺嘧啶）配对，G（鸟嘌呤）要与C（胞嘧啶）配对。理性结晶的结果是，在设计一种分子的同时，你可以预知它在X射线分析中的成像。不过这毕竟是理论，最后还是要靠实践来验证。

首先，西曼必须学会合成足够数量的DNA，这项工作花费了他在20世纪80年代初期的三年时间。接下来，他想建造一个平台，作为那些有意向的"寄宿"分子的宿主环境，如纳米电子元件或微小机器人。一个坚固的平台是至关重要的，

因为一个松散的系统无法保持DNA支架的牢固性。西曼找到了一种有效的模体[①]。这是一种双交叉结构，由联接了两次的两根双螺旋组成，而且两个联结点之间的距离很近，就像两根圆木被捆在一起，扎了两道，而不是一道。

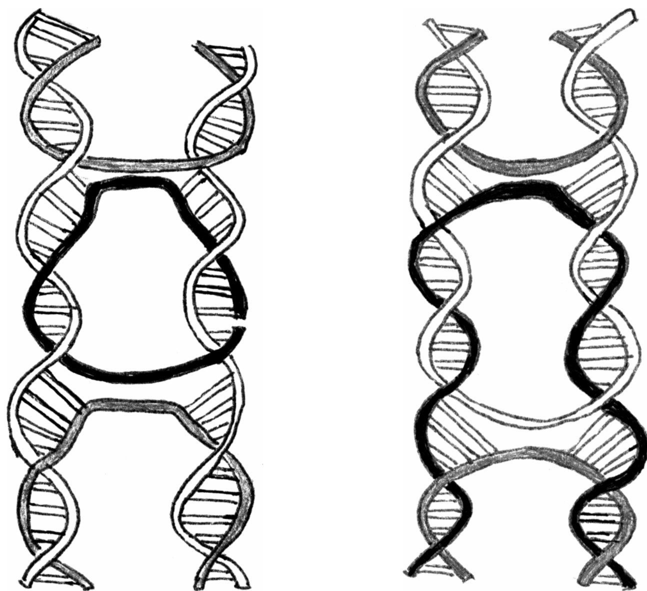


DNA链偏离线性结构的几类方式。沃森–克里克配对（A与T连接，C与G连接）可以产生出许多不同的形状（由内德·西曼提供）

这是西曼在1994年所做的发现。第二年，他的课题资助负责人建议他去参加在普林斯顿召开的第一届DNA计算大会。西曼说："如果课题负责人说你该做什么，你只要去做就是了。"在会上，西曼遇见了莱恩·阿德尔曼，后者最近用DNA解决了一个经典的七城市旅行商人问题（从原理上加以证明，参见第7章附注栏内容"阿德尔曼的DNA及汉密尔顿路径"）。他还邂逅了加州理工大学的两名年轻学生：埃里克·温弗里及保罗·罗特蒙德（见下一章）。

① 也称"基序"，指多级结构的蛋白质分子中的一些超二级结构单元，是二级结构的聚合集体。可以想象成艺术作品中一种反复出现的元素或片段，如音乐中的主题和绘画中的图案。——译者注

西曼与温弗里开始研究将双交叉结构扩展为一个二维阵列，这个二维阵列由编程生成的相间条纹组成的。接下来，西曼的实验室建造了一个双态的纳米机械设备。就在最近，西曼制作了一种小型的机器人手臂，可以在阵列中前后翻转。将两个手臂放在一起便产生了一个双臂机器人，作为DNA组装生产线上的一部分，它可以拾起其他的DNA片段。



双交叉模体的两种形式，由西曼开发，用于构建DNA的稳定的平台

一旦这些基础的部件运转起来，就打开了所有的可能性。一种可能性就是用来生产基于DNA的特制药物，通过试验这些药物的疗效与副作用，可以发现药物与潜在的药物受体之间的相互作用程度。这一过程需要高分辨率的晶体，而且对于实验室而言，还存在诸多富有挑战性的议题。在其他应用中，西曼对于用DNA来编织图案也很感兴趣，这样的编织品可以用来制造一种轻质的链甲防弹背心。

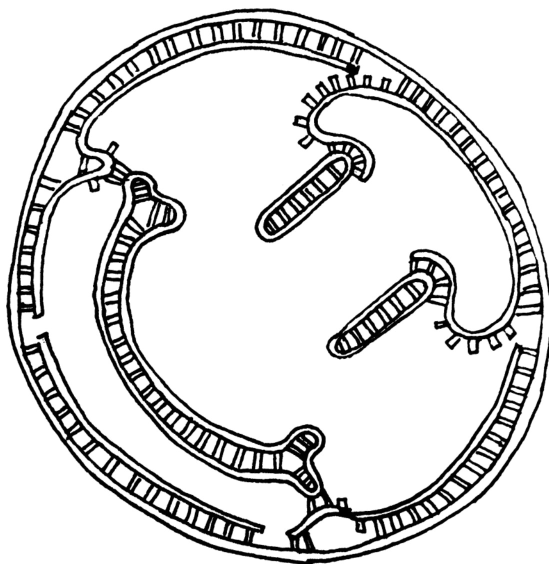
西曼的工作是坚忍的，也是富于远见的。他有的学生面对实验室工作的单调重复几乎崩溃，但还是有一些人以此为乐。与此同时，公众宣传报道却已将纳米

技术专家奉为社会名流，而西曼称这种现象为“纳米崇拜”。当他第一次参加在西雅图的纳米会议时，他和他的同事布鲁斯·鲁宾逊提议建立一个DNA支架的纳米电子系统。然而许多与会者是社会科学家、科幻小说作家和软件开发者。西曼说：“他们都是些狂热的崇拜者。”在一个晚会上，一位年轻的女性问他：“我们是最后一代会死去的人，这不是很遗憾吗？”他回答：“嗯哼？”

西曼看到了两类人之间的鸿沟，有些人满足于将科学的内涵概念化，而另一些人则整天泡在实验室里，并忍受着日复一日的重新开始与再次失败。他把训练学生克服失败心理当做自己工作的一部分：“只要你走进实验室，就很有可能会遭遇各种失败。”他还描述了化学家与计算机科学家之间在经验上的差别：对于一个程序员来说，如果知道了问题的所在，他就总能把它解决掉；相反，在化学实验室里，有些时候你想做的事情，由于受到现行的实施手段的限制，是不可能完成的。有许多情况都会导致错误，但克服障碍才能使科学产生巨大的进步，就像西曼那样。

西曼甚至建议做一个相当病态的思想实验：对科学家的长期坚守的耐力进行评价。他把这称作“异常终止测试”。他说：“设想你中途放弃了，那么你会问自己，我们还有哪些事情不知道？我们是否真的很在意？最后，这关系到要如何书写你的墓志铭。你的一生做了什么，一生的意义又是什么？并且，你是否真的做过一些有影响的事情？”







DNA折纸术对我来说是一种信心的提升。我同时有五个其他类型的项目可以做，我选择从小处做起——我不想浪费太多的钱。但我认定这是个很有价值的目标，如果能够实现，那么将意义非凡。

——保罗·罗特蒙德

第 7 章

保罗·罗特蒙德

用生命物质模仿艺术

内德·西曼已经向我们展示了如何利用“沃森–克里克配对法则”巧妙地操控DNA，假以时日，这种思路必将会流行起来，届时设计师们将基于微观的支架去建造出更多的结构。纽约当代艺术博物馆已经展出了保罗·罗特蒙德的DNA折纸术，他是费曼纳米技术奖及麦克阿瑟奖的获奖者，而他的研究才刚刚开始。

保罗·（威廉·卡尔）罗特蒙德出生于1972年，跟他的化学家祖父同名。他的父亲马克斯是一名眼科医生，而母亲朱迪丝是名摄影师。罗特蒙德在新罕布什尔州的拉科尼亚长大，这个小镇仅有16 000人，位于波士顿以北约140公里。他喜欢徒步行走在有山有水的自然美景中。他喜欢捉蛇并用书包带到学校去，这给他惹下了大麻烦。尽管如此，罗特蒙德的小学校长艾琳·赖特仍然鼓励他保持对科学和技术的兴趣。在20世纪80年代早期，学校购置了一批TRS-80计算机，但老师们却不知道怎样操作。罗特蒙德回忆说：“他们把我从二年级教室里拉出来，把我关在一间放着计算机和使用手册的小房间里，对我说：‘搞清楚怎么使用它们。’”

罗特蒙德起初想成为一名野外生物学家，但据他说，他意识到要在这个领域中做出一点成就，恐怕要花上一百年的时间。之后他喜欢上了器械潜水，并“立

志成为雅克·库斯托”[①]。但在18岁的一天，他淋了一场大雨，得了肺病，这断送了他想成为一名水下摄影师的职业梦想。塞翁失马，焉知非福。他一直觉得自己更适合做一名工程师，因为他非常喜欢刨根问底。他说：“当你面对一个复杂的系统，你认为自己对它已经非常了解了，但之后却发现，在它的背后还隐藏着某些你不知道的因素或组成部分，这样的事情会让你感到非常沮丧。反过来，在计算机和工程系统中，你对自己的所作所为了如指掌。”

罗特蒙德贪婪地阅读着理查德·费曼的书，他也是电影《天才反击》（*Real Genius*）的狂热爱好者，这部讽刺喜剧片拍摄于1985年，故事发生在一所虚构的名为“太平洋理工”的大学里，酷似现实中的加州理工学院。电影里有些非常经典的台词，如：“如果重力调转方向，你准备好了吗？我唯一搞不定的就是，怎么让兜儿里的零钱不掉出来。有了，脱光！”受到这部电影以及理查德·费曼在加州理工学院众所周知的卓越执教历史的鼓舞，罗特蒙德成了这所学校一名生物系本科生。他利用几个暑假的时间做化学研究，并在毕业时获得了生物学、计算机科学与工程学的双学士学位。

罗特蒙德的计算机科学专业的教授之一是简·斯奈普斯赫特[②]，后者师从荷兰计算机科学的先驱艾兹格·迪杰斯特拉[③]。斯奈普斯赫特不断向他的学生灌输计算中物理学的重要性。罗特蒙德至今还记忆犹新：“它的精髓在于计算的无处不在，你能想到的任何物质，都可以用来做计算，从台球游戏中的球到万能工匠[④]再到

① 法国海军军官、探险家、生态学家、电影制片人、摄影家、作家、海洋及海洋生物研究者，法兰西学院院士。——译者注

②（1953—1994）荷兰人，计算机科学家及教育家，临终前担任加州理工学院计算机科学系的行政主任。1994年2月23日凌晨，他用斧子攻击沉睡中的妻子，然后纵火自尽，烧了自家的房子。他的妻子及三个孩子幸免于难。——译者注

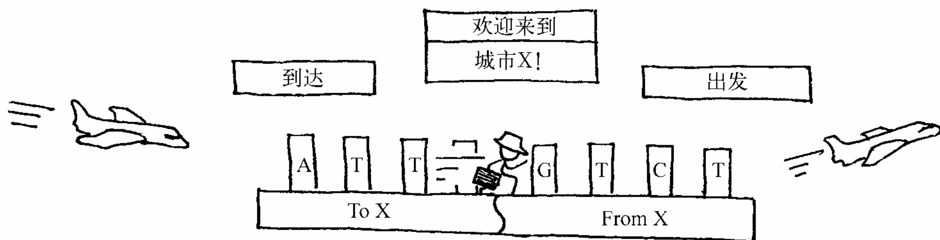
③ 关于艾兹格·迪杰斯特拉的短篇传记，收录在我们较早的书中：《奇思妙想：15位计算机天才及其重大发现》。

④ 一种发源于美国的拼装类玩具，1914年由一位石匠发明，主要部件是各种长度的插杆以及这些插杆的连接件。——译者注

DNA分子。”

1992年，罗特蒙德读到了IBM研究院查利・本内特十年前写的一篇论文，并因其中的观点感到异常兴奋。本内特提到了用DNA做计算。斯奈普斯赫特鼓励罗特蒙德，并表示像他这样同时具备生物学、化学及计算机科学知识的人，有能力解决用DNA建造计算机的问题。此后，罗特蒙德选修了亚瑟・阿布–穆斯塔法教授的课程，在完成学期课题的同时，他尝试着去面对斯奈普斯赫特提出的这一挑战。他提出了一种建造计算机的方法，即通过剪切、粘贴DNA链来模拟最经典的计算模型：通用图灵机的行为。毕业之后，他向加州理工学院的一些教授描述了自己的想法。罗特蒙德回忆说：“他们一致认为我太疯狂，他们无法给我指明方向，或给我任何建议。”

面对周围如此冷淡的反应，罗特蒙德有些气馁，他把这叫做“郁闷”。这一年的大部分时间他都在研究地理生物学，然后，在1994年的秋天，伦纳德・阿德尔曼发表了意义重大的论述DNA计算的论文。当罗特蒙德计划用一种方法去完成所有可能的计算时，阿德尔曼解决了著名的汉密尔顿路径问题的一个特例，但问题解决得非常优雅。



我们用DNA碱基序列（A,C,T,G）来编制各种符号。例如，对城市X来说，表示到达X（ToX）的符号为ATT，表示离开X（FromX）的符号为GTCT。那么toX'与toX互补，为TAA；而fromX'与fromX互补，为CAGA。这样城市X就可以表示为toX加上fromX；而城市X与城市Y之间的连线则被表示为“路程链”fromY'/toX'

汉密尔顿路径问题是一个路径搜索问题，有若干个城市（如N个城市），从一个起点城市出发，最后到达一个终点城市，沿途只能访问每个城市一次。假定路径中的每段路程都是单程飞行，比如说，从X城市飞到Y城市。阿德尔曼聪明地利用DNA对所有可能的解法进行编码，然后用最普通的实验室技术“捞出”了正确答案（更多细节参见附注栏内容“阿德尔曼的DNA及汉密尔顿路径”）。

阿德尔曼的DNA及汉密尔顿路径

作为整个过程的开始，阿德尔曼用DNA单链来表示所有城市及城市间的路程，这样，每条“路程链”的左半部分与出发城市X的DNA配对，而右半部分则与到达城市Y的DNA配对（根据沃森–克里克配对法则，A与T结合，C与G结合）。当代表所有城市及其之间路程的DNA链被混合在一起时，各个路程链左右两截分别与相应的城市链结合在一起，连缀成各种不同长度的链条，如从城市Y到城市X再到城市Z，假设确实存在这样一条路径。

阿德尔曼继续增加起点城市及终点城市的DNA分子数量。他采用一种叫做“聚合酶链式反应”的DNA复制技术，可以将一条单链放大数千倍甚至数百万倍。

他还采用了另一种叫做“凝胶电泳”的重量级技术，这种技术可以利用电路将单链及双链DNA按长度分开。他找到了那些特定长度的链，这些长度标志着这些链恰好经过了N座城市。即便这样，还会有一些链中某座城市出现两次而错过了其他城市，因此，阿德尔曼让他的溶液通过一系列的过滤器，这些过滤器用到了一些磁性球，其表面分别覆盖着与每个城市的DNA互补的DNA链。通过这一过程可以捕获到那些每个不同城市至少访问过一次的链。任何经过这种处理后幸存下来的分子，就是汉密尔顿路径。

阿德尔曼之所以选择汉密尔顿路径问题，是因为已知这个问题唯一有效的解

决方案就是遍历，即需要遍历贯穿所有城市的所有可能路径。DNA计算具有天然的并行性，这为尝试所有可能的路径提供了一种潜在的廉价方法。遗憾的是，DNA技术不能保证为大规模的汉密尔顿路径问题找到所有可能的路径。一个规模够大的问题（例如要覆盖1000个城市）意味着路径的数量将超过宇宙中已知的原子数量：做这样的实验你真的需要一大罐的DNA。

阿德尔曼的论文产生了巨大的反响，并吸引了埃里克·温弗里的注意，后者是一名研究生，也选修了阿布–穆斯塔法的课。温弗里看了罗特蒙德的演示之后，邀请罗特蒙德到加州理工学院的红门咖啡馆见面，聊聊DNA计算的事。当他们第二次见面时，温弗里提出了自组装算法的思路，即用小块DNA砖块搭建非周期性晶体的方式来模拟细胞自动机。罗特蒙德编写了他的DNA图灵机，而温弗里则编写了他的DNA细胞自动机。在1995年时，这两位年轻的科学家自费前往普林斯顿，在第一届DNA计算大会上展示了他们的论文。在那儿，温弗里和罗特蒙德会同伦纳德·阿德尔曼及内德·西曼（见第6章）共进了晚餐。

随后，罗特蒙德成了南加州大学阿德尔曼实验室的研究生，并在那里工作了6年。而温弗里则与西曼合作，但仍留在加州理工学院并担任教授。这两位朋友一直保持着合作，并延续到罗特蒙德重返加州理工学院与温弗里共同做博士后之时。在罗特蒙德的博士论文的致辞中，他称温弗里为“史诗般的英雄”，相反却称自己“更像一个丛林中的野人”。

在加州理工学院，罗特蒙德现在拥有了自己的实验室，并担任高级研究助理。几年过去了，DNA计算及结构DNA纳米技术领域差不多合二为一了。自组装算法所需的DNA砖块由西曼所发明，这一发明促成了计算机科学家温弗里与晶体学家西曼之间的一次合作，他们利用这些砖块创造出了首例二维DNA晶体。对罗特蒙德来说，这种塑造新几何结构的技艺看起来总像是一种魔法，但是到了2002年，

罗特蒙德开始用DNA砖块建造长DNA纳米管，并绕过DNA计算而直接通往DNA结构。

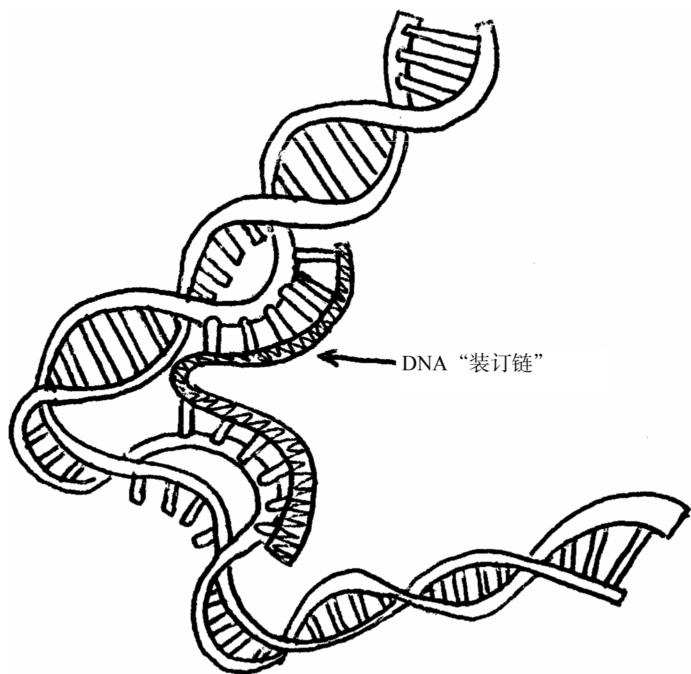
DNA折纸术源于一次与乔纳森·凯利的谈话，那时凯利还是加州大学洛杉矶分校的一名研究生，他问罗特蒙德是否能做大的重复图案及算法图案，能否打破构建最大非重复性任意结构的世界纪录。当时，西曼是这一记录的保持者，他做了大约由2000个碱基构成的被称作“截角八面体”的线段多面体。这一结构由49条链通过七步组装起来——这是一次极其复杂的技艺，因此，用同样的方法也很难超越这个纪录。

建造大型、不重复的结构需要大量独特的“化学信息”：大量独一无二的DNA序列。把它们组合在一起的过程更像是烤蛋糕：你必须确认所有材料都按照正确的比例添加。罗特蒙德突发奇想地使用了一种长单链，这样，无论是成功还是失败，都可以归咎于唯一的主料。罗特蒙德知道一种M13病毒的变种（M13mp18），它的DNA采用单链闭环结构，其基因组拥有7249个碱基（回想一下，每个碱基可能是腺嘌呤、胞嘧啶、胸腺嘧啶或鸟嘌呤）。罗特蒙德说：“有利的因素是这些病毒具有强烈的求生本能，它们具有复杂的机制，以确保在任何时候单个病毒都可以被复制，而且是被非常精准地复制。因此，这是一种应用最广泛的、完美的长单链DNA。”

现在想象你有一条DNA长链，你想把它折来折去并堆砌出一个指定的形状。你或许能塑造出那个形状，但DNA是放在溶液里的，因此可以假设你要操作的长串处于失重状态：它会到处扭曲缠绕。你可能会想到将长串中的一段与另一段扎起来，或订在一起，以确保它们彼此靠近。为了使用这种病毒来实现这样的操作，罗特蒙德提出了DNA“装订链”的概念。

每个装订链由20~40个核酸组成，平均大约30个核酸。对于M13病毒中的每个

DNA长链，罗特蒙德都用大约200个合成的DNA装订链与之混合。他添加了一点盐水缓冲液，以便于DNA形成双螺旋。他首先把溶液加热到接近沸腾，展开病毒的DNA链，然后用两小时的时间让其冷却到室温，以便让装订链将病毒的各个部分折叠成目标状态。最终他做到了。一提起他的这个诀窍，罗特蒙德就会说："真是非常非常简单。"



一个短链（装订链）可以把一条长链（本例中的病毒链）部分地"钉"在一起

从概念上讲，可以想象每条装订链都由两部分组成：左半部分和右半部分。装订链的左半部分与病毒链的某一处结合，而右半部分与一定距离之外的另一处结合。装订链之所以能够选择病毒链上正确的两点进行结合，要归功于链置换反应（参见附注栏内容"DNA配对原则"）。其直接结果是将两个有一定距离的片段拉到一起。罗特蒙德使用了大约200个不同的装订链在"埃"尺度上构建出笑脸表情。1米等于100亿埃，因此我们讨论的距离真的很短。

DNA配对原则

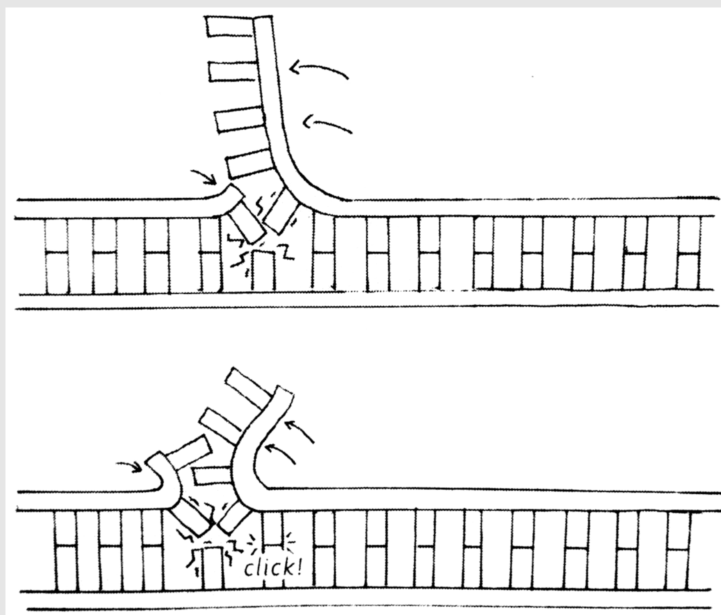
从理论上讲，每条DNA都是一个由A、C、T或G组成的字母序列。如果两条链包含一段“沃森–克里克互补字母组合”（A与T互补，C与G互补），它们就可以结合在一起。不同链上的互补片段会粘在一起，而其余的部分则悬在一旁。

例如，有一个链X＝AACCCTTTT，另一个链Y＝AAATGGGATTTTTTTT。我们可以用粗体字来强调两条链中字母序列互补的片段——如一对链包含了五个字母片段（A**ACCCT**TTT及AAA**TGGGA**TTTTTTTT），或者另一对链包含了三个字母片段（AACCCT**TTT**及**AAA**TGGGATTTTTTTT）。

根据化学反应的热力学原理，两条链会在最长的互补片段上形成稳定的结合。在一个试管中，两个五字母片段结合的几率要远大于三字母片段。通过精心设计DNA序列，科学家们几乎在任何时候，都可以按照他们期望的结合方式，精准地制作出DNA序列。

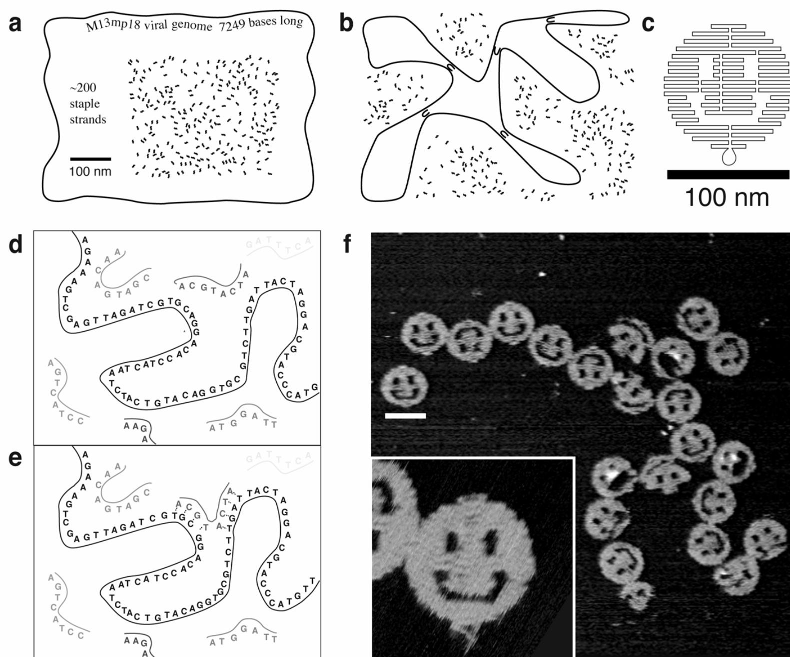
尽管科学家们的设计很完美，而在某些更适合使用天然DNA序列的场合（如用于产生DNA折纸的M13病毒），DNA链有时不能正确地按最长片段结合，而DNA“结合或未结合”的简单两态模型预示着，它们将长期保持这种“错误”的结构，永远也无法达到热力学平衡。幸运的是，DNA有办法来解决这个问题：链置换。假设一个DNA设计师想让我们前面例子中的Y链与Z链结合，Z＝TTTACCCTAAAAAAAA，这是一个完美的互补链。进一步假设，Y已经与X的五碱基片段相结合。由于Z与Y具有更长的结合片段，因此Z可以逐步贴近Y，并一次一个碱基地置换X，直到X完全脱落，即Z使得Y先与X“离婚”，再与Z“私奔”。好莱坞的影迷们对这样的情节再熟悉不过了。

伯尼·布尔克（来自朗讯贝尔实验室）率先使用链置换反应来演示了一个“DNA镊子”，其中一系列的置换反应引起了镊子的开合。现在，几乎所有的DNA纳米机械及DNA计算机都基于这一原理。



在绝大多数情况下，长的匹配（右边的部分）将替代短的匹配

罗特蒙德现在让这一过程变得惊人地容易。首先由DNA艺术家画出某种形状，也就是任务：要求这种病毒的不同的链（如X链与Y链）彼此靠近；其次由计算机程序分别算出X和Y中（在结合点处）的两个沃森–克里克互补序列；最后依据计算结果制造出一个包含这两个互补序列的首尾相连的装订链。总的算起来，一个复杂的形状可能需要几百个这样的装订链。他用电子邮件将字母序列发送给产品供应商，他们用DNA合成器来生产装订链，并用邮件把产品寄过来，每种装订链和病毒都单独存放在试管中。用机器人将它们混合在一起，并加入少量盐水，然后将溶液加热再缓慢冷却。每滴盐水中都含有艺术家作品中的500亿个实例（见插图）。如果现场就有DNA合成器，那么从概念产生到分子在试管中形成，整个过程大约只需6小时。



由DNA折纸折叠成的笑脸。(a) M13病毒长链，大约需要200个装订链来折叠它。这幅图描绘了系统在高温时，长链呈现为非结构化的闭环；(b) 在低温时，装订链开始联接长链上远端的片段，并把它们拉近；(c) 当系统达到室温且折叠结束时，长链呈现出迷宫一样的路径(装订链没有被画出)。(d、e) 示意图的特写：装订链如何联接并折叠长链；装订链ACGTACTA（上中）施加的单个约束。通常装订链长度远大于8个碱基（平均为30个），有时它们可以联接长链中的三个点，而不是图中所示的两个点。(f) 固定在云母表面的20个DNA笑脸的原子显微镜（AFM）成像。该原子显微镜用一根微型机械针来“感知”云母的表面，并拍摄折纸的图像。笑脸看起来很精巧，偶尔也会在“着陆”时断掉，大约70%的笑脸是完好的。大约1000个笑脸才相当于人的头发丝的宽度

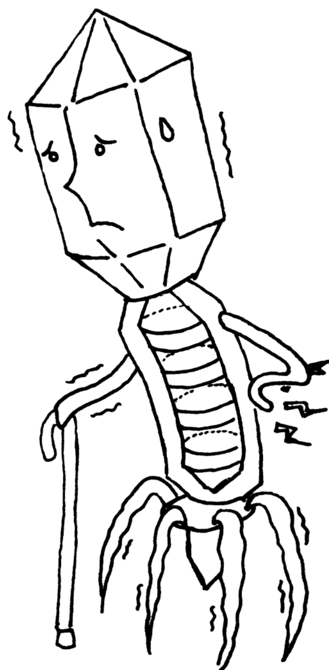
如果你觉得这件事听起来更像是编程而不是搞化学，罗特蒙德也会赞同这一说法，他呼吁道：“那是因为计算机科学家擅长思考这一类问题，我鼓励他们作为分子程序员加盟我们。”当其他研究者利用这一原则将折纸结构做得更大，同时向三维空间发展时，罗特蒙德想要采用新的自组装原则，将折纸的规模及复杂度推向极限。首先，基于DNA螺旋的特殊属性，他想用DNA折纸构成的粗糙的几何形状为元件，来制造智力联锁拼图玩具。

你可以人为地让DNA折纸的边缘呈现出DNA螺旋末端的排列，就像一面林肯积木[1]墙的整齐的末端。如果你又造了另一块DNA折纸，它的“林肯积木墙”边缘恰好与第一块折纸在形状上互补，那么这两块折纸就可以拼在一起。这种智力拼图方式可以自组装出更大型的结构。罗特蒙德最终想用这个方法与其他技术相结合来制造出裸眼可见的结构。

在其他方面，罗特蒙德和他的合作伙伴们正在探索如何使DNA折纸成为组装电子元件的平台。想想看：依据纯粹的自组装的原则，先把几管DNA混合起来，建造出DNA折纸，然后在其中加入碳纳米管，最后把DNA烧掉，就可以得到一个电路。不需要蚀刻，不需要净室，也不需要极其复杂的制造工艺。当然，如果这些在溶液里造出来的纳米电路不能被转印到材料表面，不能与常规的电子器件整合到一起，那也是无济于事的。罗特蒙德正在与IBM的阿尔马登研究中心一道，力图解决DNA折纸在硅表面指定位置的附着问题。

为电子线路构建实验电路板的想法尽管可能非常有用，但往往忽略了DNA折纸与其他生物分子实体之间的互动能力。罗特蒙德正在探索一种叫做“蛋白漫步者”的蛋白质，它可以使肌肉沿着其他蛋白质所构筑的路径做穿行运动：“我们可以用DNA折纸修建一条DNA路径，然后根据需要在沿途设置约束点，并使蛋白漫步者按要求移动。” 罗特蒙德表示，科学家可能会将DNA结构与蛋白质电机相结合，以实现对结构、计算、功能以及运动的控制。他说：“只有当我们能够再现它，而且能设计它时，才算是实现了我们的目标：有生命力的纳米技术。”

① 一种儿童积木，由带缺口的微型圆木组成，可以组装成城堡或小木屋。另外还配有合适比例的屋顶、烟囱、门窗、小动物和人物。——译者注





要研究生物学漏洞，我们可以编程实现它，来搞清楚它如何工作。或者我们能通过优化把它变得更有用。这正是计算机科学的思维方式。

——史蒂夫·斯凯纳

第 8 章

史蒂夫 · 斯凯纳

编程漏洞

病毒的核心只不过是一段DNA或RNA序列。病毒本身不是威胁，但它制造的蛋白质入侵正常细胞后，就会带来大问题。史蒂夫 · 斯凯纳和他在纽约州立大学石溪分校的同事们设计合成病毒，以期获得更好的疫苗来对抗病毒。我们承认，这多少有点吓人，但史蒂夫保证，与其说这会导致生物恐怖行为或是不可控的意外等负面结果，我们更有可能获得与病毒战斗的新武器这样的积极结果。

要理解史蒂夫的生物计算设计理念，先要懂点儿生物学。一般来说，蛋白质承担着生物体的基本功能，也是支持骨骼和皮肤的基质。蛋白质的合成需要如下两步：(1) DNA片段被复制成RNA（转录）；(2) RNA“翻译”为蛋白质。“翻译”一词很是贴切，因为核糖体这一小小的生物工厂通过读取RNA上的三联体、抓取相应的氨基酸，来装配蛋白质。然后，核糖体移动到RNA上相邻的下一个三联体，抓取氨基酸粘附在前一个氨基酸上。简单地说，蛋白质就是一串氨基酸。

DNA/RNA链被写成四个字母（A、T、C、G，在RNA中，用U代替T）组成的符号序列[①]，因此，存在64种可能的三联序列。这些序列（称为“密码子”）用

① 在第7章中讨论过，每个字母表示一种核苷酸：腺嘌呤（A）、胞嘧啶（C）、胸腺嘧啶（T）、鸟嘌呤（G）或者尿嘧啶（U）。

以编码20种特定的氨基酸。这就意味着，原DNA的数个不同的密码子可以翻译或者说是编码成同一个氨基酸。例如，GCT、GCC、GCA、GCG全都编码成最简单的氨基酸——丙氨酸。所以，GCTGCCGCAGCG这一DNA序列最终会被翻译成四个丙氨酸构成的序列。

为什么自然界允许不同的密码子编码同一种氨基酸？人们发现，对给定的氨基酸来说，某个密码子更受青睐，这种倾向性具有物种的特异性。例如，对人类来说，40%的丙氨酸由GCC三联体编码，换而言之，GCC是编码丙氨酸时最普遍的选择。相对于低频三联体/密码子来说，高频的翻译成蛋白质的速度更快。因此，用低频密码子组成基因意味着蛋白质的合成速度变慢。

如果某种病毒变异体合成蛋白质的速度很慢，那么这种变异体很可能变身为好疫苗。注射慢生长病毒让人类机体有充足的时间去识别病毒，产生相应的抗体。如果病毒复制得很快，就会引起疾病。现在，你已经了解了一点儿病毒学的基础知识，接下来，看看计算在其中如何发挥作用吧。

史蒂夫·斯凯纳生于1961年，在新泽西农村长大。他的父亲起初是收音机修理工，为传奇人物、音频发明家艾弗里·费希尔工作，当时费希尔只有3名雇员。斯凯纳说："我父亲认为未来属于电视修理，于是离开了艾弗里·费希尔，这显示了斯凯纳家族的商业头脑。"当了一段时间电视修理工后，斯凯纳的父亲成了贝尔实验室的工程师，尽管他没上过大学。斯凯纳的母亲在家照顾孩子，孩子们长大后，她开始管理当地政府的投资基金。

每年，斯凯纳一家都去佛罗里达度假，在那儿，他们经常参加回力球比赛。回力球是一种高速的球类运动，起源于西班牙巴斯克地区和法国东南部，人们用柳条编织的长勺形手套向石墙上投掷球，球速可以达到300公里/小时，几乎相当于吃了类固醇打壁球了。对于年轻的斯凯纳来说，回力球最吸引人的地方是他父

亲会给孩子们一点零钱去下注赌球，而他对计算输赢几率非常着迷。有一年，他听从当地黄牛的建议下注，赢了足够带全家下馆子吃晚饭的钱。从最初作为少年赌球手开始，斯凯纳就以联盟中八支球队的实力和比赛顺序为参数，用计算机模拟的方法建立了成功的下注系统。他在*Calculated Bets: Computers, Gambling, and Mathematical Modeling to Win*一书中讲述了这方法的原理。

赌回力球需要很多数据以及为八支队伍的输赢几率排序，而其他一些运动项目的赌博就简单得多。1977年，16岁的斯凯纳编了个名叫Clyde的计算机程序，用来预测橄榄球比赛。他甚至在本地的《新不伦瑞克新闻报》上发表赌球专栏文章。他用Clyde试着选出每一场橄榄球比赛的胜者。他说："我没有以此为生，只是告诉你这些事儿。"

斯凯纳在弗吉尼亚大学学工程专业，1983年毕业的时候，他没找到想干的工作，于是决定去读计算机科学的研究生，有康奈尔大学和伊利诺伊大学香槟分校两个选择。他觉得康奈尔大学太偏重理论，于是选择了伊利诺伊大学。有点讽刺的是，他的论文主题却是理论计算几何。斯凯纳说："除了打印，我的论文与计算机毫无关系。"

1988年，斯凯纳获得了纽约州立大学石溪分校计算机科学系的教职，在那里工作至今。1991年，他注意到他正在研究的计算几何问题和DNA的酶切过程有着密切联系，于是去找了几个生物系的同事寻求合作。他说："那些生物学家说我可以帮他们的电脑装最新版的Windows。我就跑出了生物楼，之后很多年都没再去过。"

直到20世纪中期，生物学研究还主要依靠观察和事实积累。俄罗斯数学家讲过一个笑话，科学要么是物理要么是数豆子。他们认为化学和物理很类似，而生物学左右跳不出数豆子的范围。每个豆子代表对栖息地中某种受造物的观察。对数学家而言，积累事实毫无吸引力。

在20世纪早期，生物学开始发生质变，不再只是“数据管理员”。19世纪奥地利修道士格里戈·孟德尔的遗传学研究工作，给一些连锁特性研究方面的先锋性工作带来了灵感。其中最著名的工作来自美国科学家托马斯·亨特·摩尔根，1933年，他因果蝇遗传特性的研究获得诺贝尔奖。摩尔根确定染色体是基因的载体，基因可以发生突变，而且根据在染色体上的位置，某些基因之间的连锁关系更为紧密。

虽然对遗传物质的实际结构不甚明了，但生物学家推断，相对于编码Z性状的基因，编码X性状的基因与编码Y性状的基因之间的联系更紧密。这个推理过程很简单：有X性状的生物体，几乎都带有Y性状，但并不一定有Z性状。特性的相关性使我们可以得出粗略的图谱，也就是从特性出发，找到在某种依然未知的遗传物质上的定位。

在20世纪50年代中期，詹姆斯·沃森和弗朗西斯·克里克改变了一切。基于罗莎琳德·富兰克林拍摄到的DNA晶体照片，沃森和克里克解释了DNA的结构。之后，科学家们运用DNA分析，验证了大部分摩尔根的图谱。接下来的40余年里，人们开始研究生物体中的单个基因的效应。

这一富有成效的研究导致了极端的还原论。研究者只需要关心单独的基因，控制它的行为，然后观察它的效应。典型的科学演示就是把研究的基因放在图表中心，周围全是箭头指示它的效应。

20世纪80年代后期，计算机和设备的进步使研究者们可以不再局限于单个基因，而能拓展到整个基因组。计算机和计算机科学很快就成为了基因比较和基因组合的基本工具。这些技术的发展让科学家完成了人类基因组测序，扩展了物种基因库，最后还有一点——设计活生生的生物体，这也正是斯凯纳的研究对象。他的计算机科学教育背景此时显得至关重要。

生物信息学主要是用于分析DNA序列，它将信息技术运用到分子动力学上。斯凯纳知道自己可以阅读序列，但他想做的是创造序列。这就很自然地带来一个问题（对计算机科学家而言）：什么才是最佳序列？该如何设计最佳序列？

因为不同学科知识的协作能带来新进展，所以大多数大学至少在口头上都支持这一想法：来自不同学科的科学家应当一起合作研究。知识是合作带来的一个好处，另一个好处是不同的视角。计算机科学家关心方法：找到更好执行特定功能的办法，高效储存和提取信息，解微分方程，或者提供一个逼真的计算机图形模型。但和其他学科的研究者不同的是，计算机科学家们不在意数据本身。不关注数据正是主要卖点：计算机科学家发明的方法可以广泛应用在不同领域。例如，数据库可以用在商业、艺术和空间探索上，却不需要对基本方法做多少调整。危险在于，计算机科学家可能会做出与实际应用明显偏离的假设。

考虑到这些问题，斯凯纳开始寻找生物学家合作。他的梦想是合成包含数千个碱基的DNA序列，而当时的技术水平只能合成不到100个碱基对的序列。斯凯纳想要更进一步，却找不到愿意与他合作的生物学家。2002年7月的一天，正在意大利帕多瓦度假的斯凯纳翻开了一份《国际先驱论坛报》，头版上有一幅石溪分校科学家的照片，这位科学家的办公室离斯凯纳的只有200码，看起来非常面熟。斯凯纳遇见过他好几次，只是不知道他的研究内容。这个人就是埃卡德 • 维默尔。维默尔上新闻是因为他通过定制的全基因序列，设计能自组装的DNA小片段，从零开始合成了脊髓灰质炎病毒。维默尔在合成病毒中插入了一些标记，证明所获得的是他合成的病毒而非来自其他可能的来源。斯凯纳回忆道："原则上讲，没有理由认为我们不能从零开始合成病毒，人能从试管中召唤出病毒，这个想法令人兴奋。有些人认为这听起来很危险，但全基因组合为很多激动人心的应用打开了大门。"

8

被别人抢先发表成果是科学家生活的一部分。如果被抢先了，优秀的科学家能很快恢复过来。钻研某个问题会让你变得善于解决一系列相关问题。斯凯纳认为自己的计算机科学视角能帮助维默尔优化病毒。现在，他找到了理想的合作者。维默尔可以合成用斯凯纳的算法设计的病毒并将其用于实验。美国国家科学基金和国立卫生研究院给了他们经费，去设计能引起和真病毒相同的免疫应答但毒力较弱的病毒。

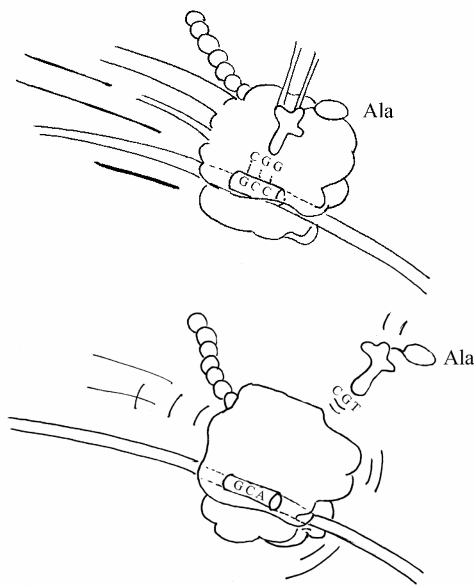
免疫应答的关键在于机体免疫系统如何识别病毒的蛋白质。如果科学家编码的病毒能产生和源病毒一样的蛋白质，那么减毒株（毒力较弱的）就可以激活与原始株相当的免疫应答。那么，问题就变成是否可以设计一个能编码同样蛋白质但对病毒本身无益的序列；它复制的速度更慢，这样就使机体能更好地清除病毒。

换句话说，他们想创造一种疫苗。当然了，脊髓灰质炎疫苗早就有了。作为前基因组科学时代的伟大发明之一，20世纪中叶，约纳斯·沙克、阿尔伯特·沙宾先后发明了脊髓灰质炎疫苗。在给人类注射减毒病毒之前，他们俩都在猴子（恒河猴）身上进行了大量实验。斯凯纳说："在猴子身上进行的大量实验将沙克疫苗和沙宾疫苗带到世间，培育的病毒更适合在猴子体内生存。"然而，其中有一个陷阱。这些从猴子身上得到的病毒与对人类致命的病毒太相似了，人体注射后，偶尔会发生致命变异。因此，沙克的方法是在实验室培育病毒，让病毒制造蛋白质，用甲醛杀灭病毒，然后将蛋白质注射入人体。这样，机体就可以产生相应的抗体。

与灭活的沙克疫苗相比，沙宾疫苗是一种减毒活疫苗。它的优势在于能提供群体免疫。你接种了，我没接种，但我可以从你这里接触到一些疫苗里的病毒，这样，我也就相当于接种了疫苗。但这又有个陷阱了。因为致病性病毒株与疫苗病毒株相比只有轻微的变异，疫苗本身可以转变为致病株（这种情况非常罕见）。卫生部门知道接种活疫苗是有一定风险的。

最终，沙宾疫苗在西方世界完全根除了脊髓灰质炎，但在亚洲和非洲的部分地区，该病仍有发生。这些年来，人们关注的问题在于如何消灭脊髓灰质炎病毒。适用于脊髓灰质炎的，也同样适用于天花及其他很多可怕的疾病。要抵御新的病毒性疾病（例如非典型性肺炎SARS）或者其他病毒性疾病有意无意的爆发，我们急需创造新疫苗。对此，斯凯纳有些想法，他问道："用其他动物携带病毒是一个痛苦漫长的过程。我们能不能设计并合成减毒病毒，避免依赖随机突变呢？"

斯凯纳把基因组看成一个程序，并由此推断：就像让程序变得糟糕一样，对基因组做点小手脚应该不难。斯凯纳和他的病毒学家团队开始寻找低复制效率病毒的设计方法。在共同发表的第一篇论文里，斯凯纳团队观察到用少见的（低频）密码子替换普遍（高频）的密码子，会大大减慢病毒的复制。想象一个颠倒的高中竞赛，奖励最慢的或是最不可能成功的选手，这样你就明白了。



我们在本章中提过，在人类等生物体内，GCC编码丙氨酸（Ala）的速度比GCA快得多，这就大大提高了生产蛋白质的速度。因此，尽管不同的核苷酸三联体可以编码同一个氨基酸，但编码效率是不同的

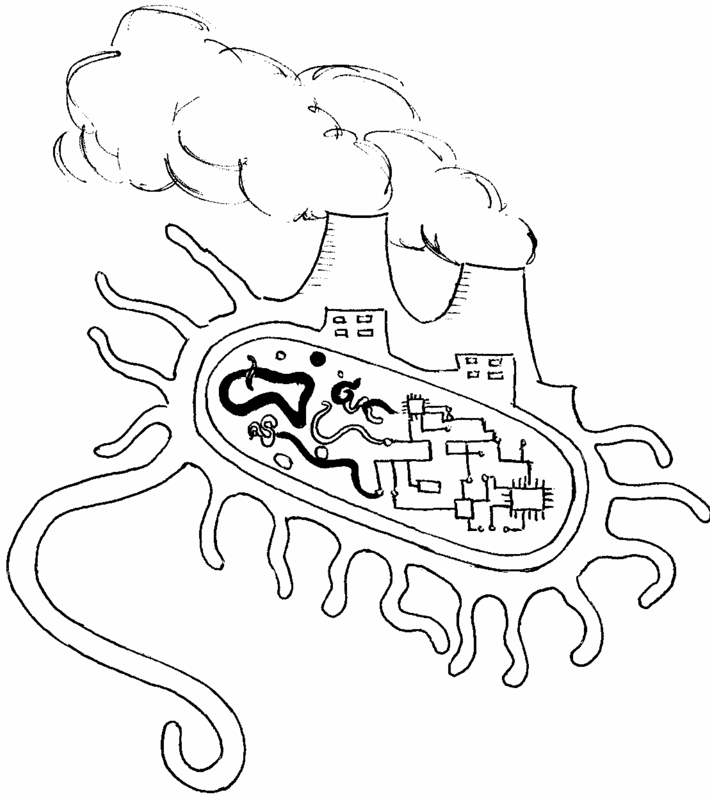
斯凯纳的工作是用低效的密码子来设计基因序列。他的病毒学家合作者们已经研究了30多年脊髓灰质炎病毒，有很多细胞能用以测量病毒生长曲线：病毒杀死细胞所需的时间。他解释说："放入少见的密码子，我们得到了慢生长病毒。"

生命科学经常在还没完全搞清楚原理的情况下，就从观察直接跳到了运用。人们熟知的阿司匹林、青霉素和奎宁水，都在还不清楚原理的时候，就用来治病救人了。用稀有密码子设计疫苗的策略也正是如此。虽然不知道为什么会存在种特异性的密码子偏好，我们还是能根据减毒病毒及其使用记录，设计出疫苗。

斯凯纳近期的工作已经不限于观察那些单个的少见密码子本身，而是进一步扩展到研究很少在这些密码子相邻位置出现的密码子，也就是说，少见的、成对出现的相邻密码子。斯凯纳和他的团队致力于找出理论依据来评价为什么有这种现象。这个问题的解决之道在于基因算法或者称诱变方法。随机的密码子序列可以尝试、诱变和评估。这个方法显示了计算机科学和生物学之间的良性循环：从生物学获得灵感的算法能帮助计算机科学家解决生物学难题。

斯凯纳和同事们的研究成果影响深远。他们能用碎片制造活病毒。然而，我们在享受这项研究所带来的智力愉悦之外，还要看到可怕的一面。怎么才能阻止有些人用普遍的密码子或密码子对来设计毒力超强的病毒呢？斯凯纳认为这个问题本身就有待商榷。他相信放入普遍的密码子或密码子对并不能获得更强的病毒。如果存在这种情况，那么野生病毒株肯定已经选择了这种基因组序列。斯凯纳说，事实上，提高翻译速度可能会让病毒株强大到对自身不利的地步。"当一个病毒翻译太快的时候，就会在还没有大量复制前杀死宿主。病毒需要生物体宿主来复制。外面的世界很复杂，但我们认为别人用我们的技术去强化病毒是极不可能的。"斯凯纳如是说。希望他是对的。







面对持续的细胞凋亡以及细胞更新和改变，胚胎发生过程中的精确性和可靠性足以让所有工程师心生嫉妒。

——杰拉尔德·苏斯曼

| 第 9 章 |

杰拉尔德 · 苏斯曼

建造十亿台生物计算机

物理学家不仅研究万物之道，还学会了一种态度：自信可以弄清一切、建造一切。毕竟，如果你的工作是研究物理法则的极限，那你也许就会想要挑战日常生活的极限。

在MIT工作期间，杰拉尔德 · 苏斯曼一直对学科界限置之不理。虽然不是个训练有素的物理学家，但因为不喜欢之前那些书里的数学标记法，他就写了本关于高等力学的书。因为已有的工具表达能力太差，他就创造了自己专属的硬件设计工具。在加州理工学院的天文学家的帮助下，他建造了“数字太阳系仪”去解决一个自牛顿时代起就悬而未决的关于行星轨道扰动的问题。而现在，他想用细菌造个有十亿处理器的机器，因为他认为现代电子设计已经过于复杂和昂贵了。这一概念的灵感来自胚胎发生：细胞分裂并创造新生命的过程。苏斯曼的态度始终如一：值得尝试，而且他会带着极度的乐观和幽默感去努力。此外，他还是“书呆子骄傲”（Nerd Pride）活动的发起人之一。

苏斯曼生于纽约布鲁克林区，是药剂师的儿子。他毕业于伊拉兹马斯霍尔高中，这是一所位于弗拉特布什、拥有众多名人校友的名校。国际象棋冠军博比 · 菲舍尔、演过《活宝三人组》的莫 · 霍华德都是这所学校著名的退学生。著名的毕

业生则有芭芭拉・史翠珊、尼尔・戴蒙德，还有毕业于1964年的苏斯曼本人。高中时期，苏斯曼是个热心的修补匠，喜欢拆收音机，做点新玩意儿。他一直都保持着修补匠的热情。大约25年前，他开始制作手表。他俏皮地说："三十岁之前，人缺少耐心。六十岁之后，人的眼睛就不行了。"

十几岁的时候，任何关于技术的书他都读。1956年，约翰・麦卡锡和克劳德・香农合著了具有开创性的*Automata Studies*一书，内容是关于现代计算机基础的图灵机和细胞自动机的。受此书的感召，苏斯曼决定去MIT。在那里，他先后于1968年和1973年获得了数学学士和博士学位。虽然研究的是数学，但他的生活中充满着计算。在缺少计算机的20世纪60年代，这种全神贯注显得非同寻常。

刚进大学的时候，一个师兄告诉他科技广场里有些没人用的计算机，包括一台数字设备公司（DEC）的PDP6。DEC PDP6是数字设备公司最早的大型计算机之一，只为大学定制。师兄告诉苏斯曼说："你可以去看看，玩玩。"有这一句话就够了，他成了技术中心的常客。他回忆道："有一天，一个秃头的家伙出现了，我当时以为他要把我扔出去，因为他显然是老板，而且都35岁上下了：一个老头子。"这个"老头子"正是马文・明斯基，神经网络领域的先锋。明斯基没把他扔出去，而是问他是否愿意来打工编程。震惊之余，苏斯曼急忙答应了。

明斯基问苏斯曼为什么他的井字棋程序是随机的。苏斯曼说因为他不想让程序有任何预设的误解。明斯基回答道："哦，它有的，你只是不知道究竟是什么。"苏斯曼对此印象深刻："这是我听过最聪明的事儿啦。"苏斯曼说自己从此"和这个家伙亲近起来"，余下的MIT学生时代，他一直为明斯基工作。

1951年，明斯基用一个随机连线的神经网络机奠定了神经网络领域的基础。他的想法是创造一个网络模拟生物神经网络，可以根据刺激改变连接。几年后，明斯基和约翰・麦卡锡一起，开创了人工智能领域的先河。跟随他们的脚步，苏

斯曼的博士论文聚焦于机器学习。

拿到博士学位后，苏斯曼还不确定自己接下来想做什么。他很喜欢为明斯基工作，但不知道自己的职业生涯该如何发展。谈到当时的情况，他说："有一天，几个MIT的人说：'我们听说你很不错，要不要来当个教授什么的？'"苏斯曼征求了明斯基的意见。按照苏斯曼的说法，明斯基告诉他说："这是个糟糕的想法：教授责任太多却没啥权力。"但苏斯曼的妻子让他还是去试试看。

1973年，苏斯曼拿到MIT的教职，在那里工作至今。他的办公室设在计算机科学和人工智能实验室（CSAIL）。这是MIT最大的跨专业实验室，有800多名成员。CSAIL目前位于弗兰克·盖里设计的颇具争议的建筑物里，该建筑物2004年启用，名为斯塔塔中心，又名极客宫殿（Geek Palace）。角度不同的不规则墙体和金属光泽，让斯塔塔中心看起来似乎就要倒下、破碎或者扭转。

1975年的一天，苏斯曼参加了MIT同事卡尔·埃维特的讲座，正好与哈佛本科生盖伊·斯蒂尔坐在一起。埃维特的发言很有煽动性，但很难懂。苏斯曼和斯蒂尔决定展开进一步讨论，后来变成通宵编程试图实现埃维特的想法。很快，他们发现自己重新实现了拉姆达演算（详见附注栏"拉姆达演算：简史"）。

拉姆达演算：简史

拉姆达演算是20世纪30年代由阿隆佐·邱奇和斯蒂芬·克莱尼发明的一种数学标记法。它是1958年约翰·麦卡锡发明的前卫语言LISP（list processing的缩写）的基础。[①]除了大量应用在人工智能外，主要由于它递归的概念，LISP还影响了其他一些编程语言。麦卡锡发明LISP是因为他想计算代数表达式的微分。如果你还记得高中时学的微积分，表达式的求导有时牵涉到子表达式的求

① 在我们之前的一本书《奇思妙想:15位计算机天才及其重大发现》中有麦卡锡的传记，还讨论过他发明的LISP。

导，这里就有递归。

要理解递归，想想下面这个关于祖先的定义：如果某人是我的父母或者我父母的祖先，那么此人是我的祖先。这个定义好像绕了个圈，因为祖先既是要定义的，又是定义的一部分。但计算机可以把这个定义变成完全有效的过程。给定亲子信息，并以我为起点，我的祖先是我的父母及父母的祖先；我父母的祖先是他们的父母和他们父母的祖先；一直继续到没有更多的父母信息为止。

拉姆达演算中的拉姆达（λ）让程序员可以定义一个函数而无需命名。比如我们在高中都学过，定义$f(x)=x+3$。这就是说，名为f的函数接受一个参数x，返回一个比x大3的结果。因此$f(5)=8$。在拉姆达演算中，我们可以记为$\lambda x.x+3$，因此，$(\lambda x.x+3)5=8$。不用命名函数非常方便，特别是当它们由程序产生时。苏斯曼和斯蒂尔解释了如何使用拉姆达表达式编程以及如何高效执行。

苏斯曼和斯蒂尔开发出了Scheme语言。该语言强大的功能和极简的风格使它成为最受欢迎的编程教学语言。苏斯曼令人信服地论证了用Scheme语言编程，学生们能学会由牛顿建立的研究气体、固体和行星运动的高等经典力学。他认为用Scheme语言编程比让学生推导数学公式要有效率得多，因为传统的数学标记法有些模棱两可。

林恩·康韦研究超大规模集成电路。上了她的课之后，斯蒂尔发现Scheme语言还可以用来设计能执行Scheme语言的硬件。1979年，苏斯曼、斯蒂尔和杰克·霍洛韦、艾伦·贝尔一起制作了一台直接运行Scheme语言的计算机。用语言创造一个芯片来运行该语言，在MIT校园里激起了反响。到了1980年，MIT的导游们开始向参观者们展示可以自我复制的机器人。人类提供部件（不是人体部分），而机器人自己完成余下部分。

一些技术人员认为这一切只是开始。1994年，马文 · 明斯基在《科学美国人》上发表了著名的文章，其中写到因为我们的生物组成部分（肝、心和脑）寿命有限，所以机器人或者机器增强人将会统治地球。苏斯曼持不同意见。他不研究硅基计算增强的生物有机体，而是设想基于生物组件的计算。生物组件杂乱无章，难以控制，但相对于制造硅芯片所需的清洁房间和复杂机器，生物组件有一个极大的优势：它们几乎是免费的。

“设想你有这样的计算机，沙粒大小，成桶买回来，搅拌成混凝土；你能通过每秒百万次的浮点运算得到混凝土。假设你能找到办法给它们供电，那能否在上面编程做些有趣的事情？”苏斯曼问道。

苏斯曼认为大量的实体不会有任何特定的局部模式，但在三维空间上，它们可以通过*无定形计算*和周边的邻居交流。1994年，苏斯曼的研究生安迪 · 柏林写了关于沿梁放置传感器防止不锈钢梁屈曲的博士论文。如果传感器感应到屈曲，它们能通过控制一个机械装置（就像有电流就收紧的“肌丝”）来阻止屈曲。只用了一点点电力，柏林就让不锈钢梁的强度得到了重大改进。

苏斯曼的朋友爱德华 · 弗雷德金、诺尔曼 · 马戈利斯和托马索 · 托福利建立了一个竞争学派，制造并实验了*细胞自动机*。自动机原本是由数学家斯坦尼斯拉夫 · 乌拉姆发明的，20世纪70年代，约翰 · 康韦又将它简化成名为生命游戏的二维棋盘游戏。想象一个无限大的棋盘，在时间*t*，自动机根据细胞自身和周围细胞在上一时刻*t*-1的值，为棋盘上每一个细胞分配新的值。康韦给出了一个分配的规则。弗雷德金、马戈利斯和乌拉姆设计了更多规则，并探讨这些规则在物理世界中的应用。近年来，为了在他的书*A New Kind of Science*中讨论科学的许多分支，斯蒂芬 · 沃尔弗拉姆大大扩展了这一套规则。

基于三点，苏斯曼不同意这种方法。第一，细胞自动机的栅格太规则，比任

何生物系统都整齐得多。第二，细胞自动机假设了同步性——棋盘上所有的方块同时更新，这在真实的生物细胞中不会发生。第三，没有为故障做准备。在活着的生物体中，细胞会死亡、突变、变形。

尽管杂乱无章，但真实的生物系统表现近乎完美，这让苏斯曼惊叹。他谈到制造蛋白质的核糖体。他说："核糖体是巨大的分子机器：很多分子用一种复杂的方式结合在一起。"从功能上说，每一个核糖体都很完美，都完全一样。苏斯曼相信科学能找到办法让核糖体做点别的事情。苏斯曼和他的研究生们——丹尼尔·库尔、拉蒂卡·纳格帕和罗恩·韦斯，还有曾经的学生、长期的同事汤姆·奈特——已经开发出一种编程语言，可以建模并能最终控制细胞生长。

苏斯曼对自然进化的长期目标很好奇："木材是一种奇妙的东西，但并不完美，它有结节之类的东西。那么，做出完美的木材不是很好吗？木材中的纤维素是由木质素胶合在一起。如果我们能让某种细菌分泌肽聚糖，一种淀粉状分子，用来粘合木材，难道不是很有意思吗？"木质素可以让细胞壁坚固，因此，木质素越多木材越耐用。全部分子都排列整齐的高木质素木材应该既牢固又轻巧，并可能得到新的木材结构。苏斯曼承认他的团队远未达到这样的目标。他说："我只是在想象30年后的事情。"

实现这样的梦想很不容易。要理解这种挑战，就想想看工程师希望从材料中得到什么：控制和可预测性。电子电路正具有这两种特性。材料位置稳定，物理性质为人熟知而且能数学建模。与此相反，生物体的细胞湿湿地、蠕动着，还会死掉。每个细胞都与众不同，这么不可靠的东西似乎啥也干不了。但是，胚胎发生——细胞分裂、复制、分化的过程——证明控制细胞是有可能的。细胞本身并不完美，但成群的细胞极为精确地形成了鼻子、皮肤和心脏。问题在于如何把这种现象变成工程规范。

苏斯曼的同事汤姆 · 奈特发明了生物砖的概念，来建造细胞计算的大厦。他的灵感来自电子工业。电容、电阻、晶体管必须用铁、一点儿金和其他特殊合金组成。电子工业用不同的材料创造出通用组件。奈特在基因水平上做了类似工作（参见附注栏“生物砖入门”）。苏斯曼说：“生物砖技术是关于如何制造出能通过有趣的方式连接在一起的标准化部件的，一切才刚刚开始。”

生物砖入门

生物砖计划的目标是创造一个能连接在一起的电路模块的仓库。元件库里应该有专用的软件工具，以便设计者在计算机上规划生物电路，并在放入生物基质之前实验该生物电路的行为。元件将放置在细胞内，由细胞提供能量。生物砖采用人工的基因启动子对的形式。目前，生物砖被设计成既不干涉细胞的正常行为（别玩火自焚），也不干涉其他生物砖。设计者们已经得到了第一个数字化的生物砖，来证明他们克服了细胞内的噪音。

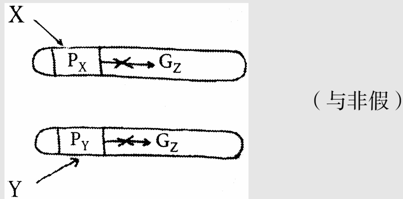
要了解工作原理，你需要知道一些细胞生物学知识。DNA由生产蛋白质的**基因**组成。每个基因的右边都有**启动子**（基因的“上游”，因为DNA有方向性）。基因要生产蛋白质，上游的启动子就必须绑定诱导蛋白质而不是抑制蛋白质。某些启动子在没有抑制蛋白质粘附的情况下，也能启动下游基因的**表达**（转录成RNA，然后生产蛋白质）。它们不需要诱导蛋白质，只要没有抑制蛋白质就行。我们称这种启动子为“缺席”。大多数基本的生物砖是由缺席启动子和附着基因组成的**换流器**。这个换流器能让基因只在不存在抑制蛋白质的情况下进行表达。也就是说，当抑制蛋白质无，换流器电路打开，反之亦然。

你需要知道的下一个分子生物学知识点是任何基因启动子对都可以合成。也就是说，任何基因都可以粘附在任意启动子后。假设启动子P*x*是缺席启动子，能被蛋白质X抑制。进一步假设基因G*x*表达蛋白质X。这样，就可以将P*x*结合在G*x*上，得到一个负反馈电路。当X存在，P*x*被抑制，导致G*x*停止表达。结果，

9

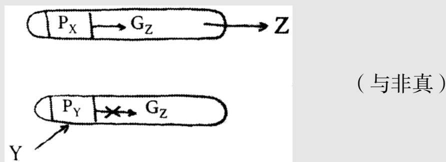
X逐渐减少，Px又开始诱导Gx表达更多的X。这种效应就是周期性表达。

X、Y同时存在，Z被抑制



（与非假）

X、Y有且仅有一个存在



（与非真）

如果蛋白质X、Y同时存在，蛋白质Z就不再生产。
如果仅X或Y存在，那么其中一个基因启动，Z就会生产

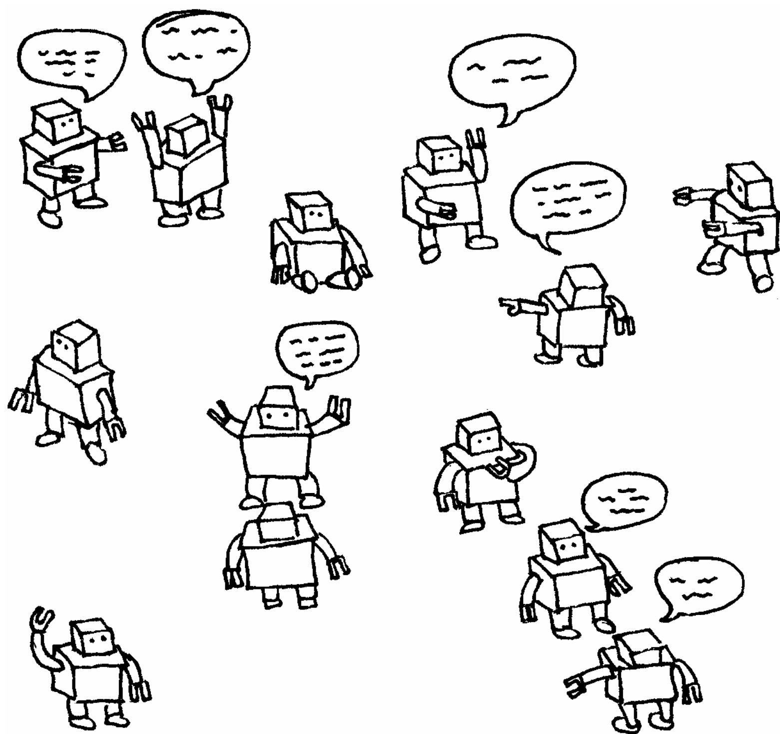
还有其他一些可能的电路。例如，你可以将Px与Gy相连，得到一个数字电子设备：与非门。与非门电路可以生成任何包含与、或、否的逻辑功能。当两个输入至少有一个为0时，与非门都能产生一个为1的输出。首先，通过结合两组启动子基因Px-Gz和Py-Gz的设计，可以得到一个与非门。当蛋白质X、Y同时存在，Gz就不表达，因此就没有蛋白质Z。在其他任何情况下（X、Y仅存在一种，或都不存在，即为0），都会产生Z。

换流器和与非门都是细胞内电路。外部信号（来自人类的）能到达这些元件，而且细胞之间能够交流。特定的诱导分子在细胞间扩散，并由于浓度差而进入细胞内。诱导分子改变特定抑制分子的行为，避免抑制分子对缺席启动子的干扰。这样相互联系的效应只需要一个开关。

这项研究还处在起始阶段，但元件已经有了：电路元素、能量源和交流。生物学是一团乱麻，如果要控制细胞，那么就要更多地和细胞交互。但这毕竟是一个好的开始。

如果可以创造出标准化的生物部件，那么它们会非常小，直径仅几微米（百万分之一米）。它们可以堆在一起培养，也许还能计算。当然，它们会发生故障，苏斯曼相信生物砖的容错能力比电子元件强。他说："生物有一个非常好的特性，就是当环境改变，动物或植物就产生进化，这就是内在的进化潜力，你一定得做些什么让它们能在失败中成长。同时，你也不想让它们发生你不要的改变，你要的是可控的适应性。"如果能从生物材料中获取这种强韧性，计算就能变成免费的，至少接近免费。因为细菌数量庞大、能移动，因此除了计算之外还可能在工程上发挥作用。苏斯曼说："我想用细菌做工程基础设施，想让细菌成为制造设备：让它们变成车间。未来，我还想用细菌制造分子结构。"

可能有人会觉得我们会被某种吞噬一切的灰色粘液灭亡。苏斯曼丝毫不担心这种事情："通常，对技术的非议是说我们并不了解所有事情。我们不能解决这个问题：确实是不能什么都知道。"苏斯曼相信什么都不做更糟糕："一般而言，自然进程就像无人驾驶的汽车。有时候，比如说二叠纪大灭绝，就是这辆车撞树了。和以前那些物种不同，我们可以握住方向盘。现在是不知道怎么开车，但我们可以学。如果不试试，总有一天，我们也会跟其他东西一样撞到树上。没有什么事儿能持续几百万年。"



■ ■ ■ ■

我不想写什么神秘的程序。我要通过编程将局部和整体联系起来，这样，最终你会理解发生了什么。

——拉蒂卡 · 纳格帕

| 第 10 章 |

拉蒂卡 · 纳格帕

从局部到整体

现代数字计算机的标志之一就是信息的最小片段，字节，要么是1要么是0。另一个显著特征很少被提及：“正确性”假设。软件程序员假定硬件能正确工作。通过检测电路和设置冗余，硬件设计者能补偿和隐藏硬件故障。在假定硬件能正确工作的前提下，程序员就只需要考虑软件的正确性。

现在，想象一下你在计算上千甚至数十亿作为传感器嵌入在大桥上或人体内的生物细胞或机电细胞等实体。单个细胞可能在无预警的情况下发生故障。怎么才能用上这一大堆不可靠的组件，特别是那些只和少数相邻组件通信的组件呢？想过要怎么编程吗？这就是哈佛大学工程和应用科学学院的拉蒂卡 · 纳格帕研究的问题。

1972年，纳格帕出生在亚特兰大，她父亲当时在乔治亚理工大学攻读机械工程博士。8岁时，全家回到印度北部的阿姆利则市（Amritsar）。纳格帕喜欢上学，特别喜欢艺术、数学和生物。高一时，她父亲买了台早期的英特尔芯片个人计算机去管理他工厂的账目。他用BASIC语言写了个程序去存储账本。女儿的工作就是录入数据和查找漏洞。有一天，她正在录入数据时突然停电，数据全丢了。阿姆利则经常停电，纳格帕就写了个程序来保存数据，免得再发生数据因停电丢失

的情况。然后，她又自学了写屏保。就这样，她被计算机编程迷住了。

在20世纪80年代中期的印度，人们通常希望理科成绩好的女生能成为医生，家人也希望纳格帕能走这条路。但她决定不当医生。她说："当医生，人们带着一大堆问题来找你，这一点儿意思都没有。我自己的问题就够多了。"

高中快毕业时，阿姆利则成了恐怖活动的目标。1984年6月，锡克教圣地金庙被印度军队攻击了。作为报复，同年10月，英吉拉·甘地总理被她的锡克教侍卫暗杀。形势越来越糟。政府开始宵禁，如果在宵禁时间走出家门，纳格帕说："那就只能听天由命。气氛真的是非常非常压抑。"虽然经历了这样的创伤，纳格帕依然与印度锡克教文化保持了紧密联系。她在当地传统舞蹈彭戈拉团体中非常活跃，还为一个供印度海外移民追寻传统文化的网站Chowk画了插画。

当纳格帕考虑上大学的问题时，她最希望的就是远离印度压抑的气氛。因为她成绩优秀，又是美国公民，所以，她可以去美国上大学。1990年，MIT接受了她的申请，她决定去那儿学习工程。

作为MIT的新生，纳格帕有一门课是计算机科学导论。教材是由哈尔·埃布尔森、盖瑞·苏斯曼及其夫人朱莉·苏斯曼合著的《计算机程序语言的结构和解释》。因为该书英文版的紫色封面上画着巫师，所以有时候又被称为"巫师之书"。自1985年出版以来，这本书就成了MIT导论课程的圣经。纳格帕回忆道："它非常漂亮，内涵深邃。我第一次知道计算机能做什么，我们能用计算机做什么。"

纳格帕全力以赴学习计算机科学。她爱编程，并参加了每一次编程竞赛。她还要上物理和数学必修课，这些课程内容她在印度都学过了，所以觉得很无聊。她说："计算机科学是我的救星。"MIT的计算机课程非常重视基础。纳格帕回忆道："入门很慢：电路如何构造存储器？电路如何做加法？电路如何产生时钟频

率？你得从这些基础学起，突然你就造出计算机了。”从简单的元素构建复杂的东西，每一层都建立在下层基础之上，然后用模块创造出更多东西，这让她着迷。

当年，MIT有一个让学生在合作公司获得硕士学位的计划。纳格帕参加了贝尔实验室的计划，当时贝尔实验室还很是欣欣向荣。她参加了雷・麦克莱伦和艾伦・贝尔鲍姆领导的一个课题，为早期掌上电脑制造低耗能、低速的处理器，比如苹果公司的Newton和Eo个人通讯器。纳格帕说：“作为一名学生，在那种环境里真是令人兴奋。他们还不习惯跟年轻人一起工作。因此，在那里实习，所有这些名人都会跟你说话，你会听到他们在计算机科学领域30多年的经验之谈。”

纳格帕在贝尔实验室看到人们享受生活、创造发明，连业余爱好都充满智慧。1995年，纳格帕回到MIT读博士。她原本打算研究网络，但很快就发现不太自由，因为互联网的模式根深蒂固。她说：“我觉得我想要更自由地探索。我想去一个全新的领域，在那里可以发明出新东西。”

纳格帕本科期间的好朋友丹尼尔・库尔，是CSAIL“无定形计算”研究组的创始人哈尔・埃布尔森和盖瑞・苏斯曼的学生。库尔鼓励纳格帕来他们组。1996年，组里刚刚完成了第一篇关于无定形计算的白皮书，库尔拿给纳格帕看了。她说：“它真的就在眼前。”她觉得他们能用无数小装置创造未来。它是并行计算，没有拓扑结构的限制，有传感器和促动器能与物理世界相互作用。纳格帕找埃布尔森和苏斯曼谈了谈。她回忆不同风格的两人，说道：“盖瑞告诉你全景，而哈尔告诉你要重点研究什么。”整个团队正在尝试让成千上万小而简单的装置做些有用的事情。这些装置只是以差不多而非精确相同的速度运行，只跟最相邻装置通信，还经常发生故障。

胚胎发生过程中，即使有些细胞出错，胚胎依然可能形成。过程复杂，但一般来说结局都很圆满。胚胎发生过程极为稳定，否则，没有物种能繁衍生息。稳

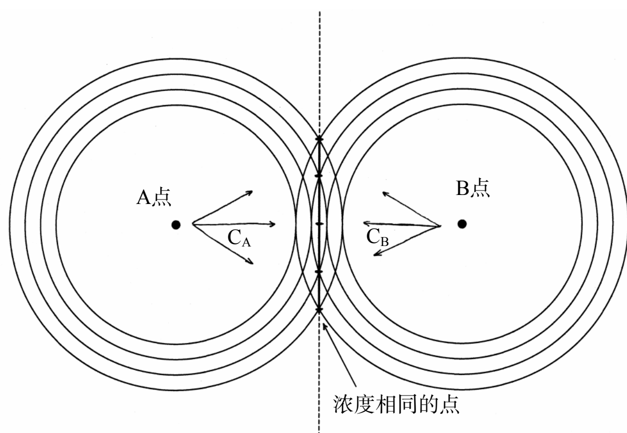
定性也是斑马条纹等模式形成过程中的一个问题，即有小错误的细胞如何在不同频率（相当于计算机术语“不同的主频”）下复合在一起形成条纹。

事实证明，生物是最节约的。物种虽多，机制却只有几种。其中一种机制就是化学梯度。通过接收到的化学信号强度，细胞可以“分辨”自己和化学物源头的距离。化学梯度仅需要局部通信：化学信号从一个细胞扩散到另一个细胞。相应地，扩散将无定形计算面对的主要挑战之一转换成了一个优势。数以亿计的细胞可能个个形态不同，但它们的普遍特性却是非常统一的。

所有与化学信号源等距的细胞，即使信号通路上的某些细胞出错或是某些细胞接受信号较慢，它们接收到的化学信号量依然基本相同。这种现象提示我们，在不同步的微观元素基础上对宏观特性进行编程是可能的。例如，在不同时间激活的微小细胞可形成直条纹。

诺贝尔生物学奖得主克里斯蒂亚娜·尼斯莱因–福尔哈德曾在《科学美国人》上发表过一篇文章，提出了一种机制。她指出，在果蝇早期发育中化学梯度负责精细模式的形成这一机制非常简单。模式形成本身特别有趣：重复利用相同的原理创造复杂的模式。之后，纳格帕读到了彼得·劳伦斯和路易斯·沃尔珀特在分子和发育生物学方面的一些研究成果。胚胎中简单的理念可以导致复杂的整体行为——从形成不同大小组织的能力，到部分再生的能力。这深深地吸引了纳格帕。

要了解这些原理，就设想一下你要在A和B两点之间画一条黑线（线段AB的垂直平分线）。方法很简单：在同一时间，让A释放化学信号C_A，B释放等量的化学信号C_B 。将中间接收到等量C_A和C_B的细胞标为黑色，结果会得到一条接近线段AB垂直平分线的黑线。如果起始的量不同，黑线就会靠近化学信号释放量较小的那一端。



线段AB垂直平分线上的每一点都接受等量的化学信号C_A和C_B。中点上的点接受两种信号的量都最多

纳格帕将不可靠细胞的无定形点编程方法归功于丹尼尔・库尔的“生长点语言”，并称其令人“大开眼界”。他演示了如何在计算中利用各个点上简单的梯度去构建所有模式。构建模式是一种挑战，但在工程中，模式必须转化为形态。这正是胚胎所做的。纳格帕解释道：“这不仅仅关于模式。模式只是过程的起点，之后，模式能导向变化。”

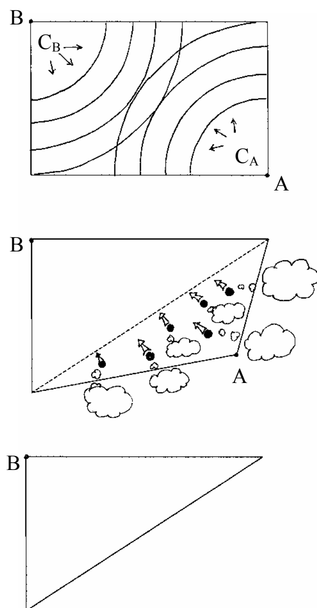
促动器为形成折叠（生物折纸的一种）提供了可能。日本数学家藤田文章证明，通过四种方法可以折叠出任何形状的折纸：(1) 折叠两点之间的线段；(2) 把一个点折叠到另一个点上（因此，两点间垂直平分线上有条折痕）；(3) 将线段的端点折叠到一个点上；(4) 折出射线夹角的角平分线。几乎可以折出任何形状的折纸，由此，纳格帕想到也许可以用一种折纸型语言描述胚胎发生：“你可以用一种整体的方式来考虑。”折纸为计算机科学家们提供了*抽象层*的概念。一个好的抽象层提供了灵活又简洁的语言去描述期望的输出（在这里指形态），对底层的计算基质（在这里指生物或机电细胞）也是可编译的。

在纳格帕的术语中，折纸基元是整体，细胞基元是局部。她的想法是允许程

序员在整体层面编程，然后翻译器将其转化成在局部层面执行的代码。相比于直接把整体目标映射到每个个体的行为，纳格帕的做法将问题一分为二：(1) 实现整体目标；(2) 将构建步骤映射到局部规则。

让我们看看细胞如何从矩形折叠成直角三角形。假设细胞能移动，右下对角点A释放化学信号C_A，左上对角点B释放等量化学信号C_B（如图）。每个接受C_A比C_B多的细胞都向B移动（通过向相邻细胞移动，获得更多C_B）。细胞可以一直移动到某一点，在该点，细胞接受等量的C_A和C_B。结果，所有细胞都从A侧，沿着垂直平分线移动到B侧并均匀分布。

在纳格帕的折纸形状语言（OSL）中，她选择了另一种不同的方法。她假设细胞不能移动，而是一层细胞共同作用变形。因此，如果细胞沿直线协作、收缩，就能让整层折叠，拉近远处的细胞。



矩形右下角的细胞接受的化学信号C_A比C_B多。这种浓度差就是一个信号，让细胞向高CB浓度方向移动，这样就形成了直角三角形

通过细胞移动或者细胞层移动，都能用微观计算元素实现宏观形态变化。这为创造可编程材料提供了潜力，但纳格帕想得更远。生物细胞感知环境，这种能力能帮助我们做编程设定。

纳格帕和她的团队已经设计出了模块化的桌子，在不平的地面上，桌腿可以变长或收缩来保证桌面水平，每条腿都能感知平衡并做出响应。这种智能桌的理念是感知和改变形状从而适应环境。她梦想能让断臂智能化，在愈合过程中适应不同的压力。如果传感器和促动器能嵌入桌腿，也就能够植入人腿，就像电影《机械战警》中那样。

纳格帕希望能在高度抽象的水平给数以千计的传感器和促动器编程，让它们工作起来。给大量传感器和促动器编程能让机器人协作形成所谓的“超级机体”，正如博物学家爱德华・威尔逊描述的蚁群和多细胞生物一样。纳格帕想放一小群机器人在丛林环境下，观察它们是否能像蚁群那样生存。她评论道：“人们谈到杀手级应用时，常常是在谈论‘拯救世界’。但他们不知道这一切如何发生。当这种技术实现了，很多聪明人就可以说：‘嘿，我能用它解决问题。’然后，突然间就有了无限可能。”



| 第三部分 |

物理和速度

1958年，第一个集成电路（或称芯片）发明以来，芯片上集成的晶体管数量呈指数增加，每两年翻一番。自1965年戈登·摩尔首次预测了这种现象后，这种趋势就被称为摩尔定律。处理器的速度也变快了，从Intel 4004上市的1971年到2005年间，处理器的速度也几乎是每两年翻一番。这种令人欣喜的结果来自于小型化和更快的主频。

计算机主频就相当于振荡频率固定的节拍器。例如，2千兆赫主频就是1秒钟内振荡20亿次。主频速度决定了单个处理器的性能：每振荡2次，处理器执行一个指令。按照数学家约翰·冯·诺依曼发表的结构来看，每个指令都从计算机存储器（内存、缓存或者寄存器）中抓取数据，执行操作，将结果返回给存储器。因为硬件进步如此之快，程序员们知道在下一代处理器到来的时候，他们的程序会运行得更快。这真是个好消息！

但还有个坏消息：提高主频速度也导致了能耗的增加。数年前，主要芯片厂商发现继续提高主频速度所需的电力过大，会危及电路的完整性。厂商们认为最好的方法是在不提高主频速度的同时，利用更小的晶体管和更窄的导线宽度，在芯片上集成多个处理器，这也就是所谓的“多核处理器”。为了最大限度利用这类机器，编写和支持并行编程成为必需。

蒙蒂·代诺一贯坚持设计最快的计算机。他的哲学是让问题主导设计的几何结构。他和他的小团队已经设计出了每秒千万亿次计算机，几乎比目前的笔记本电脑快一百万倍。这种计算机可以用来解决物理难题或是破解密码。针对一类特定问题进行优化后，代诺能实现最佳性能。他想象中的架构是缩短数据移动距离来提高性能。想象一下，你能告诉客户让产品在两年内有百万倍的改进，而你只需要几个人的团队吗？代诺已经做到了。

从很多方面来看，戴维·肖的理念和代诺很接近，但是他只关注一个问题：蛋白质的分子动力学。他用一个或几个分子的三维原子模型作为起点，然后计算当原子相互作用施力时，原子如何运动。这就要求在足够短的时间内移动模拟的原子，以避免近似错误——所谓足够短的时间大约是1飞秒（千万亿分之一秒）。蛋白质做令人感兴趣的事约需1毫秒，1毫秒等于1万亿飞秒。每一步都需要计算数千个原子间的相互作用。肖和同事们已经设计出一种机器，模拟这种霎那间发生的步骤。

代诺和肖都试图设计数字电子器件来解决极具挑战性的计算难题，数字电子器件遵从这些难题背后的物理学。然而，数字电子器件未必是最好的硬件媒介。蛋白质折叠只需1毫秒，但就算是肖的宝贝计算机安东（Anton），模拟1毫秒的折叠也需要好几天。为了避免运算错误，数字电子器件输入内存的值只能是0或1，但构成数字电子器件的晶体管和电阻能产生各种电压值。有些科学家找到了利用这一事实的方法。

尝试给自然界的难题建模时，乔纳森·米尔斯更愿意使用连续的物理学，而不是离散的电子。自然界中的连续问题包括力场中粒子的运动和天气模式等。在数学家李·鲁贝尔研究成果的启发下，米尔斯造了“扩展模拟计算机”，它能解一些表征很多物理难题的微分方程。它本身包含了一种其物理特性可以描述成多种

微分方程的材料（主要是些泡沫）。通过编程，让这种材料模拟问题，再测量材料的输出电压，就能解决一个物理问题。米尔斯发现相较于通过大量计算来模拟微分方程，这种方法更为自然。他的研究结果还处在初级阶段，但很有说服力。

斯科特·阿伦森追求的是截然不同的范式：量子计算。尽管数字计算机中的每个字节都只有一个状态（0或1），而一个量子比特（qubit）可以部分是0部分是1。这种不可知的特性可以扩展到很多量子比特，20个量子比特能编码超过百万个20字节的组合。如果能编码出问题的每一个可能解，从中选取最优组合，那么就能高效解决一批重要但未知的问题（称为NP完全问题）。这种可能性尚不明朗，但阿伦森决心攻克难关。我们对自然的理解（包括时间旅行的可能性）可能都要靠量子计算了。





观察单个处理器在单一时钟周期下在做什么并且出手改变它，这多少有点上帝的感觉。它被根据工作任务设计得十分完美，这种感觉就好像设计出恰到好处的赛车引擎般。

——蒙蒂·代诺

第 11 章

蒙蒂·代诺

速度缔造者

各行各业都为计算机的体系结构所吸引：数学家和工程师，甚至音乐家和视觉艺术家。之所以涉足这个领域，通常是因为他们直觉地知道如何借助熟悉的领域理解概念问题。但几何仅仅是个蓝图。计算机架构师像建筑师一样，必须全面考虑问题，解决电力传输、散热和故障处理等问题，还必须考虑成本。和建筑师的不同之处在于，计算机架构师要拿出几乎不会出现故障的产品，同时还得承担采用新技术带来的风险。

蒙蒂·代诺在位于纽约市约克镇高地的IBM沃森研究中心工作，是新的千万亿次计算机的架构师。所谓千万亿次计算机是指计算机每秒可以执行千万亿条指令。蒙蒂的设计来源于数学哲学。回顾他的成长经历，很有意义。

代诺生于1948年，在新泽西州的林肯公园市长大。他的父亲起初是一名汽车销售员，后来在工厂上班。二战期间，代诺的妈妈在一个飞机设计厂上班时，整天都在用机电计算器。战争结束后，男人们回到职场，他的妈妈就没再得到过类似的工作，而只好去了工厂上班，有时候还有夜班（孩子们因此学会了自己做饭）。他的父母都崇尚教育。代诺回忆道："我妈妈要我以后当医生。她甚至让我去学拉丁语，因为她觉得医生一定要懂拉丁语。"

代诺的求学之路并非一帆风顺，多半是因为他在课堂上总是不安分。甚至到现在，谈起他的超级计算机，这位资深设计师依然怀着极具感染力的热情。但小时候，这种无法约束的能量给老师们带来了很多麻烦。幸好，他爱上了数学。他的数学老师是爱沙尼亚人，她让代诺坐在班级后排的角落里，这样其他人上课时，代诺可以学他自己的东西。代诺说："从某种意义上说，她拯救了我。要不然，我就完蛋了，我会虚度光阴，惹上麻烦。"那时候他还开始鼓捣电子学，主要是无线电。

代诺后来进入了MIT，但并没有按照父母的想法去读医科。他先是迷上了哲学：本体论和神学。他形容自己是个"笨小孩"，成长缓慢，而且只愿意按照兴趣行事，不考虑现实状况。最后，他还是去学了数学。代诺很遗憾自己错过了老师的指导："我确实不够成熟。我被扔进宿舍，和一批人相处，其中至少一半人有阿斯伯格综合征。"

代诺获得了波士顿大学数学逻辑学硕士学位，然后去伊利诺伊大学继续学业。他的博士导师盖西 · 塔库提给了他一个难题，需要用到无限维度的希尔伯特空间。希尔伯特空间在量子力学和信号处理中非常重要，它将几何空间从二维、三维延伸到无限维度。此前，这方面的问题没有什么人研究。代诺是否能解决它，没人知道。但有一点是很清楚的：解决不了，他就拿不到博士学位。代诺说："回想起来，为啥我要冒这个险呢？但事已至此，我只好努力努力再努力，并终于解决了它。"冒险和辛苦并存，这一点贯穿了代诺的整个职业生涯。

代诺认为计算机科学是工科，直到读研时，他才开始上第一次编程课。这门课教授的是BASIC，一门基本的编程语言，他很喜欢。他去计算机工程班学习设计电路，并参加了一个比赛，他设计的电路在上百人的作品中脱颖而出。从此，他真正开始了计算机架构师的生涯。代诺颇惊讶于自己的胜利，因为他没有任何

工程背景，而且自认为依然是个“搞哲学的”。那时，他开始建造自己的计算机，甚至设计显卡。1978年，在获得数学博士学位的同时，他还得到了计算机科学硕士学位。

代诺号称自己是“混”进IBM的。不管事实是否如此，他的第一个项目——布线机，就证明了自己的价值，这是20世纪80年代初基于64位微处理器构建的第一批并行处理器之一。

每个数字芯片都由通过电线连接的晶体管和电阻构成。如果两条电线接触，就会短路。如果将所有线路都放在一层平面上，那么互连的可能性就很小。因此，芯片有好几层（类似楼房的楼层），这带来了更复杂的布线模式和极具挑战性的布局难题。IBM要求代诺为公司的高端机型找到最佳的线路布局。

代诺用64个Z80微处理器组成了一个8 × 8的网格。他想让处理器和将被布线的芯片通信：每一个处理器只负责芯片上的一小片区域。处理器之间借助穿过不同区域边界的线路进行通信。通过这种方式，Z80处理器基本上能够独立、并行地工作。代诺从这个项目中学到的重要一点就是：“计算要与工作匹配，这个需求必须满足。”

当时的IBM靠着巨大的单处理器计算机统治世界，人们连双处理器的想法都并不急切，使用64个处理器看起来更是鲁莽。代诺回忆道：“每个人都说这太荒谬了。你不可能用64个处理器同时协作处理一个问题。幸好，我们没有停手，而是把它做成了。”

为了将64个处理器连接在一起，代诺遇到了两个让所有并行机设计师挠头的难题。首先，是可靠性：机器总是在崩溃。然后，要编程让一堆叽叽喳喳的处理器干活，而不是互相闲聊到死。要让计算正常进行，他要确保每个Z80处理器能

独立完成大部分必要的工作，并且只是偶尔跟其他的处理器通信。

布线机展示了新范式的前景：并行机可以比大型机运行得快得多。讽刺的是，二十年后，这一范式开启了微处理器群组取代昂贵的大型机的时代，这让IBM濒临倒闭。

代诺的下一个机器被称为“约克镇模拟引擎”（IBM沃森研究中心位于约克镇高地）。制作芯片耗资巨大，而且一旦错了就不得不推倒重来。因此设计约克镇模拟引擎就是为了在芯片正式制造前，模拟它的逻辑操作。当时还有很多其他的软件模拟器，但代诺的要快几千倍。一个巨大的交换机与很多计算机主板相连，模拟信号从一个电路发送到另一个电路的过程。代诺说：“这么大的风险，只有在IBM才有条件实现，好在我们得到了上千倍的回报。”

一般来说，提高速度有两个办法：好算法或者好机器。某些问题显然很适合用并行机来解决。例如，设想一个问题有很多种独立的交互场景，每一种都必须尝试。这种情况下，并行机中的每个处理器都能尝试一个场景而不影响其他处理器。完成一个场景后，处理器可以接着处理另一个。比如说，金融公司用并行处理研究不同利率对债券价值的影响。

有些问题则需要偶尔的互动。打个比方，地方政府的工作与此类似。绝大部分城市规划问题，比如土地使用，都与其他城市无关。但诸如交通之类的问题就涉及一个县、州，或者区内的很多城市。尽管市长们基本都是自己做决定，但偶尔还是要和身在别处的同事们有所交流。代诺的布线和模拟主要用到本地的资源，但也有一些问题是全局性的，面对着极大的工程挑战。

20世纪80年代中期，一些物理学家请代诺设计能用量子色动力学理论做预测的计算机。量子色动力学描述胶子、夸克等亚原子粒子间的相互作用。相关理论

是否正确还是未知数，但用实验来验证理论预测是很容易的。问题在于如何计算出预测。

算术计算是层层推进的，要获得新的数据必须等待前一步计算的结果，说不定需要5个时钟周期才能完成计算。代诺这样说道："开始计算，然后你就得坐在一旁傻等。"这种被称为管线停滞的延迟，会让很多机器无法发挥全部性能。另一个难题是存储。计算机在读取存储的同时，并没有满负荷工作，因而没有得到充分利用。第三个问题在于通信。计算机通过数据包（由0和1组成的二进制数据单元）通信，这么多数据包在传输时会为了挤占网络而打起来。

要解决所有这些问题，就需要代诺称为"编译计算"的方法。量子色动力学预测中要重复执行很多相似的操作，编译计算能很好地解决这一问题。编译意味着利用计算的某些特性，只在必要的时候移动特定的数据。代诺学会了怎样把计算机调节成像蓄势待发的赛车引擎一样。他能建造每秒执行110亿条指令的计算机，并将其命名为GF11，这种计算机比当时的商用机型快几百倍。

20世纪90年代，代诺试图换个思路，解决通过传输线发送连续完整图像的问题。他试图重构图像，而不是发送压缩的像素。要想高效地实现这一想法，就需要计算机快速执行被称为射线追踪的计算机图形算法。代诺指出："问题在于射线追踪在所有机器上都运行得很糟，尤其是我们自己的机器。"当时的计算机是为了将很多块数据移动到缓存线而设计的。除非能利用所有数据，不然性能就很差。缓存线是一种存放临时数据的区域。

射线追踪可以逆向使用，还原射线射入眼睛的路线。用几百万像素的面板代表眼睛的光感受器，为了确定它的颜色和饱和度，可以用射线追踪法假想从瞳孔射出的光线。射线击中红色就停住，这个像素就应是红色；如果射线被反射，就会继续前进，并且得到更多信息。在这个场景中，到处都可以收集数据，但路径

无法预测。因此，每道射线都必须从存储器中的多个部分收集数据。代诺意识到因为射线追踪无法定位，所以在常规计算机上无法实现。

代诺开始设想射线追踪计算机的模样。他认为处理器要“简单、有点缄默”，完成一个操作就乖乖等着。一个处理器是否空闲无关紧要，因为它在硅晶片上只占一点点地方。代诺把这些处理器称为线程单元。每个线程单元负责一道射线。不幸的是，就和其他美妙的新技术发明经常遇到的那样，这个东西没有市场。没人在意远程会议时图像是不是逼真。好在只要你抓住机会，一流的技术总能找到新用途。

有一天，代诺在他所在楼层的复印机旁巧遇一位同事。这位同事述说了一个称为蛋白质折叠的问题，还说现有的计算机都太慢了，没办法计算这个问题。他问代诺射线追踪计算机是否可以做这个。代诺稍微调整了一下原始设计后，意识到自己的机器在蛋白质折叠问题上会做得很好。这个机会出现在20世纪90年代后期。代诺和他的团队完成了一个方案，这就是后来著名的“蓝色基因”项目。他说：“我很自豪能想到这个名字。”但是，IBM的高层有些担忧，因为此前他们从没制造过有160个处理器（线程单元）的芯片。大多数人认为这么做毫无理由，也不认为它属于正式的IBM架构。

后来，一位在保密地点的“不便说明的政府客户”对这个项目产生了兴趣。代诺的方法不仅可以用于射线追踪和蛋白质折叠，还可以解决很多重要问题，这一点已得到证明。代诺从没告诉我们他的客户是谁，我们也知道不能问，但很多密码破解机构对高性能计算机似乎总是得陇望蜀。例如，美国国家安全局从20世纪50年代起就资助了很多IBM的项目。

实际建造的时候，蓝色基因计算机还是采用了每个芯片上两个处理器的相对保守的构架。即便如此，在蛋白质折叠、量子化学等应用中，它依然达到并保持

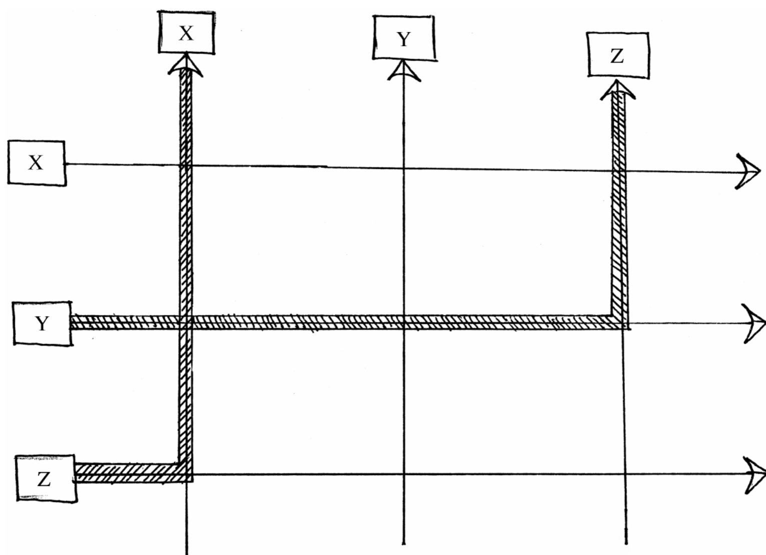
了每秒500万亿次的速度（每秒进行500万亿次数值计算）。

政府资金支持代诺制造了代号为“独眼巨人”（Cyclops）的线程单元机。代诺关于缄默处理器的观点从根本上简化了处理器设计。我们的笔记本电脑里是些“聪明的”处理器，它们会猜测下一个指令是什么，还得准备好猜错时的备用方案，并针对文本和数值过程建立一整套指令。这种范式需要上百个设计师和很多电力。相比之下，代诺把很多简单的处理器放在一个芯片上，每个处理器在某时刻只执行一个指令。事实上，这种处理器简单到可以共享“浮点”硬件（对分数执行算术操作）和快速存储器。

代诺预言独眼巨人将是第一批千万亿次计算机中的一个。他的团队人数极少，他自豪地说：“只有通常规模的5%。我有一些极具天赋的朋友和学生，他们是独一无二的群体。”

计算时间大多被这么多处理器之间的通信延迟所消耗，于是代诺选择了交叉开关，它看起来像个大棋盘。每一个处理器X都在X行X列上。处理器X与Y通信时，X发出的信息沿X行送达Y列，再转至处理器Y。这种结构将每对处理器之间的通信分成两步，但需要非常多的线路（技术上讲，是处理器数量的平方）。代诺的同事格雷戈里·齐纳斯基（代诺称他是自己认识的“最聪明的人之一”）解决了芯片上布线的问题，尽管这让项目又延期了好几个月。

容错是一个更大的问题，而且随着电路组件的体积越来越小，问题愈加凸显。α粒子和宇宙射线轰击小的电路元件会导致出错。因此，代诺设计了冗余。独眼巨人有两块并列的芯片，能进行同样的计算，但客户不愿意付双倍电费。要知道，电力消耗可是以兆瓦计的。代诺团队设计的替代方案是让计算机进行可自校验的计算。计算粗糙不要紧，可以再算一遍。有些情况下，确实需要再多算一次。代诺说：“这就是我们现在的处理方法。对未来而言，这种方法可能好，也可能不好。”



可以同时处理分别从Y行到Z列和从Z行到X列的两条信息，因为它们既不共享行也不共享列

故障有可能发生在开机之前。制造芯片需要超净房间，因为即使是一粒灰尘，都会毁掉芯片。独眼巨人使用的芯片是IBM有史以来制作的最大的芯片。芯片越大，产量越低。也就是说，制作100块芯片，只有10块合格（10%的产量）。芯片越大，可能出问题的地方也就越多。

怀疑情绪再次在人群中蔓延。代诺有一个推理：为什么全部的160个处理器都必须完好？有158个好的又会怎样？换芯片既消耗时间，又让人觉得非常挫败。他说："越碰它，它越容易坏。"独眼巨人就这么绕过坏掉的处理器，坚持运行。代诺认为他的容错方法能得到广泛理解："它会普及起来，因为这种大芯片总是供不应求。"

代诺的客户们都热爱追求速度。他的下一个计算机可能会达到几十倍于千万亿次的速度。眼下的一个应用是支持大型射电望远镜的数据处理。但眼前的成功说明不了未来。实际上，代诺说："我们没法比现在更快了。"他指的是物理法则

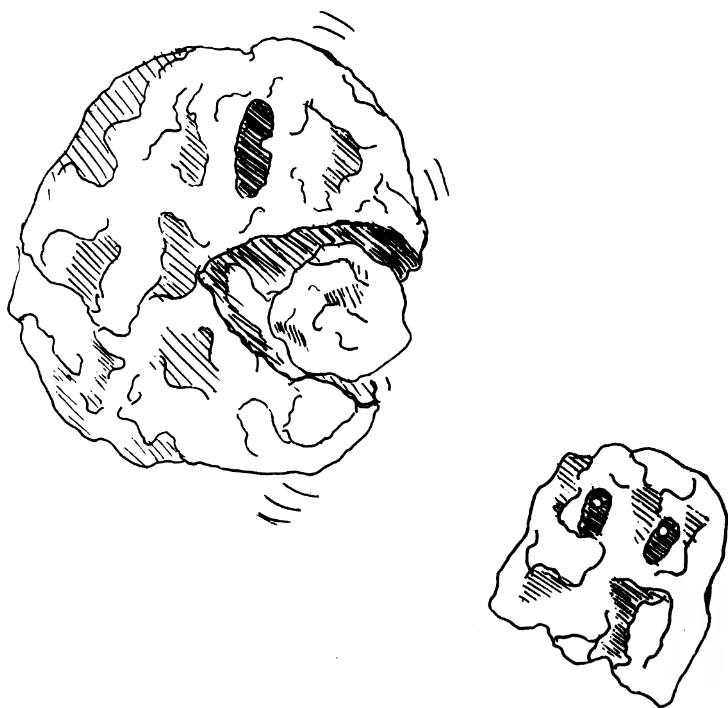
的限制。为了节能，输送到芯片的电压比以前要低。电压降得过低可能会让芯片错误地把0变成1或者把1变成0。另一个问题是芯片之间的带宽。代诺很看好直接用光学连接（例如激光）取代线路。在那之前，代诺说："我们有麻烦了。"

对于只专注一个目标的单一应用，代诺已经知道如何把通信降到最少，用自校验计算管理故障，节能并散热。他说："这里面的另一个难题是整合软件，简直是受罪。他们还像30年前那样编程：用C和C++。工具太差了。"

代诺认为软件的惰性来自于集体的惰性。如果没有其他人用，那么就很难让人们使用创新性的语言。事实上，他认为至少在科学计算方面，软件开发倒退了。谈到APL语言时，他说："很神秘，看着很像腓尼基语。一个矩阵会占据整个屏幕，然后问题就解决了。"回想一下我们对杰克 · 洛夫莱斯研究工作的讨论（详见第4章），一小群富有创造性的程序员使用极具表达能力的阵列语言，其中很多理念就来自代诺提到的APL语言。考虑数字的整个阵列，而不是单个数字，程序员能更简洁地表示问题。例如，用K语言表达最短的数独程序仅需要不到100个符号。

阵列语言的简洁不光对程序员有好处，对编译器也是如此。编译器是一段程序，能将编程语言转化为机器能够理解的代码。当程序员在一个操作中声明"让两个包含一百万个元素的矢量相乘"，编译器就将此任务分配给多个并行处理器。如果程序员使用C语言家族中的"标量"语言，那么编译器就得挨个检查每一个元素，需要耗费很多时间去确定任务分配。在数据控制操作中，也有同样的限制，例如，"找到所有年龄在40岁到60岁之间的雇员"。将一个操作应用于大量数据的能力，推开了并行计算机性能显著提升的大门。很快，每台计算机都将是并行机。

语言学家爱德华 · 萨皮尔和本杰明 · 沃夫因研究语言影响人类思维方式和能力而闻名于世。计算机语言同样影响着计算机。为代诺的大型并行机编程的程序员们需要一种能让数以千计的处理器开足马力的语言。





最重要的是为了将来拯救生命做出自己的小小贡献。结果未必尽如人意。也许我们会了解一些科学上有趣的事情，但没人知道它对药物发明能起到多大的影响。

——大卫·肖

|第12章|

大卫·肖

安东和巨大的飞秒镜

“理性的人适应世界，非理性的人一直试着让世界适应自己。故而所有的进步都依靠非理性的人实现。”这是乔治·肖伯纳的名言，完全可以用来形容大卫·肖（两位肖先生并非亲戚）。

大卫·肖最好地阐释了“非理性”一词。他在研究起步之初就已经拥有旁人难以企及的条件。他在1988年成立的德劭（D. E. Shaw）集团为他带来了巨大的财富，而且彻底改变了数理金融交易的性质。公司的理念是用了不起的人，提供杰出的技术保障，利用金融市场的低下效率来挣钱。

肖快50岁的时候，偶然与一位年长40余岁的前辈交谈了一次，前辈劝他将自己的定量和计算才能用在与金融完全不同的领域：在原子水平上研究生物系统的行为，这一领域称为分子动力学。肖当前的目标是将这一领域内的问题解决速度提高1000倍。

肖1951年生于芝加哥，成长在洛杉矶。他从很小就喜欢科学。三岁时，他梦想要成为医生。肖回忆道：“度过了几年肢解盆栽和沙丁鱼罐头的日子，我就要求父母给我买一把外科手术刀，让我在地下室解剖老鼠。”肖的父亲是等离子体物理

学家，母亲是艺术家并拥有艺术教育博士学位，他们在肖13岁时离婚了。肖的继父是加州大学洛杉矶分校管理学院教授，他让肖了解了研究者的生活方式。

肖在加州大学圣迭戈分校完成了本科学业，主修数学和应用物理。上了神经网络领域先驱大卫·鲁梅尔哈特和实验心理学家唐纳德·诺尔曼的课之后，肖对认知的计算模型产生了兴趣。在两位老师的建议下，他去了斯坦福大学的人工智能实验室读博士。斯坦福人工智能实验室是一个生机勃勃的地方，研究内容涵盖人工智能领域的各个方面，包括机器人技术、计算机视觉、机器学习、语音识别和自然语言理解等。

读研中途，肖休学三年去开了一家从事软件开发和咨询的小公司，之后又回到斯坦福完成自己的博士学业，师从吉奥·魏尔德侯尔德和特里·温诺格拉德。肖的课题是用新型大规模并行机加速某些类型的人工智能和数据库管理应用程序。这完全是理论研究，而且肖从未建造过任何硬件。当时，肖对计算机系统的实际面貌几乎一无所知。

1980年获得博士学位后，肖得到了哥伦比亚大学的教职，去领导一个项目，并按照他博士论文中提到的机器，装配可用原型。系统使用由肖的实验室设计定制的集成电路。电路包括很多小处理器，每一个小处理器都有自己的存储器。通过在同一个芯片上集成逻辑和存储，机器就避免了冯·诺依曼瓶颈：在处理器和存储器间传输数据所需要的时间。

冯·诺依曼瓶颈限制了传统计算机系统执行程序的速度。肖用的基于知识系统的“非冯”机是高度并行的。它并非通用或者专为科研设计的超级计算机。当时，西摩·克雷正在设计这种具有开创性的超级计算机。而“非冯”机针对的是复杂模式匹配问题，这些问题涉及通过“意义”搜索，而不是通过简单的关键词。

完成了“非冯”原型机后，肖对开发用途更广的大规模并行机产生了兴趣。尽管从学术标准来看，“非冯”机得到了很多资金，但肖很快意识到开发大规模并行机的资金预算远远超过一直资助他的政府基金数额。他想可能是时候寻求商业资金了。然而，拿出商业计划书后，他发现风险投资者对并行处理的初创项目毫无兴趣。

搭建自己机器的同时，肖和几个机构谈过，其中就包括摩根斯坦利投资银行。当时，该公司刚发现股票市场的某些异常，而且开发了一种能自动利用银行自有资金进行高收益股票交易的计算机程序。为了彻底开发这种新交易类型的潜力，摩根斯坦利正在寻找和肖有一样背景的人，将新技术引入它进行中的研究。此前，肖从未考虑过华尔街，但他对在全球金融赌局中运用技术感到很兴奋。另一个诱惑是摩根斯坦利开出的薪水：6倍于哥伦比亚大学教授所得。肖回忆道：“我意识到跳槽能让我很快赚到钱，让家人过上舒服日子。”

1988年，肖离开摩根斯坦利，并创立了德劭集团。他招募了一个学院派研究团队，致力于在全球金融市场寻找投资机会。当时，数字金融学家已经完成了量化交易的基础研究，少数投资者也已经实验了基于相对简单的方程的系统策略。

德劭集团作为新一代投资公司的雏形，提供高级量化和计算技巧，用方程阐述构造复杂的大规模证券投资组合。目标是系统化控制各种风险，以持续获得高回报。肖是华尔街第一批“量化投资专家”，交易记录辉煌。他回忆道：“公司起步的时候，计算金融学几乎还是一片处女地，所以到处是唾手可得的果实。”

当公司雇员超过千人，管理的投资基金超过400亿美金时，肖发现自己花了太多时间在管理上，几乎没有时间从事技术工作。他回忆说：“我想做真正的科学，为了让我这部分脑子高兴，我开始在晚上或是洗澡的时候，研究一些随机应用数学问题。”

肖的朋友、哥伦比亚大学化学教授理查德·弗里斯纳开始给肖一些他在结构、动力学、蛋白质功能等研究中遇到的计算小问题，从此，肖的消遣找到了侧重点。肖之前曾从赛·利文索尔那里了解到蛋白质折叠问题，肖在哥伦比亚大学当计算机科学老师时，利文索尔是生命科学系的系主任。利文索尔知道肖在制造大规模并行机，想知道是否能模拟蛋白质折叠。虽然这些问题引人入胜，但经营德劭集团让肖无暇抽身。

是否要重返全职科研领域，肖深思了好些年。2001年，肖50岁，在和比尔·戈尔登共乘一辆车子去参加生日宴会时，当时已91岁的戈尔登鼓励他追求梦想。年轻时，戈尔登也在华尔街赚了大钱，然后离开，投身于自己的梦想：将科学知识带到政府最高层。戈尔登促成了美国国家自然科学基金的设立，并说服杜鲁门总统设置总统科学顾问，在核武器、太空竞赛、环境和能源政策等一揽子极具挑战性的问题上，科学顾问们提供了连续性的行政指导。60多年来，戈尔登是包括肖在内的很多初出茅庐的美国科学和技术政策制定者的导师。1994年，肖被克林顿总统任命为总统顾问委员会科学技术顾问。

肖说："当比尔·戈尔登说'去做吧'，我开始认真考虑了。"科学是肖的最爱，而且他训练有素。他现在可以自费做研究，这令人羡慕。戈尔登的建议和人生轨迹，打动了肖，肖自己出任德劭集团的首席科学家，并且不再管理公司的日常金融业务。

肖觉得蛋白质动力学问题非常符合他的兴趣以及自己开发新型计算机架构的背景。此外，还有个个人原因：他的父母和姐姐都死于癌症。尽管没受过医药方面的科研训练，但他想设计一种计算工具，并期望这种工具有朝一日能用于开发拯救生命的药物。他说："我不幻想成为抗癌之战的将军，但当个战士也很不错。"

分子动力学检测多原子结构在自身环境中的构型，进而推断构型在原子间力

的作用下，如何随着时间推移而改变。蛋白质通常有上千个原子。此外，周围还环绕着水分子。目前尚无可靠的公式来预测——摆弄方程解决不了问题，只能靠模拟。

进行分子动力学模拟，需要把时间划分成非常短的时间步骤，差不多每步1飞秒。1飞秒是10^{-15}秒，即一千万亿分之一秒。在每一步里，要计算出所有原子间的力，然后再算出这些力作用的结果，即每个原子在下一飞秒的位置。这些步骤要重复很多遍。

对肖来说，“很多”意味着上万亿次，因为大多数有趣的生物活动（例如，蛋白质折叠或拼接）都发生在毫秒内。如果傻乎乎地去计算（这是脑海中迸出来的第一个办法），那么每种情况下都需要计算每个原子对其他每个原子的影响。换句话说，虽然只有几万个原子，这个过程却需要计算蛋白质–水分子对中上亿次的相互作用。传统处理器可能需要好几秒去模拟1飞秒，模拟万亿步就需要超过10万年。大规模并行机能解决很多问题，但这个问题中含有一个基本的顺序：必须等这一飞秒结束后，下一飞秒才能发生。挑战就在于让这种基本的飞秒级模拟速度加快。肖的研究重心就在于此。[①]

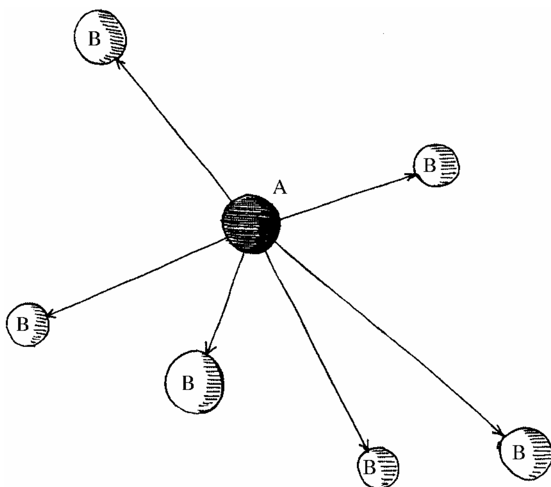
好在有几种方法能显著减少远距离原子的计算需求。虽然细节不同，但每种方法都对相距很远的原子做了精确的近似。就算是用了这样的技术，依然必须计算每个原子与全部相邻原子间的力。问题在于如何算得更快。

如果在并行机里，每个原子的信息定位在一个处理器，我们就能想到好几种

12

① 推断蛋白质或蛋白质复合物的结构还有一种方法，就是用已知的蛋白质结构（蛋白质X射线晶体结构分析）数据库来帮助预测未知的结构。例如，不同蛋白质中特定的短氨基酸序列通常以相似的方式折叠。有一套非常成功的软件Rosetta（是西雅图的华盛顿大学大卫 · 贝克实验室开发的），就体现了这种方法。在好几个竞赛中，Rosetta都成功预测了蛋白质折叠，而且这在制药业中的应用也开始普及，最终可能会发展出一种结合了经验和分子动力学的混合策略。

方法。其中最自然的方法可能是将原子A的信息发送到每个相邻原子B的位置（如图）。用肖的术语来说，就是将A的信息发送到每个相邻B的“领域”。相对地，A能在自己的领域里接收来自每个相邻B的信息。但这个方案依然需要移动很多数据。肖的方法借鉴了这个思路，但让从A到B的信息传送发生在“中立”领域，既不在A的位置也不在B的位置（参见附注栏内容“中立领域的计算”）。



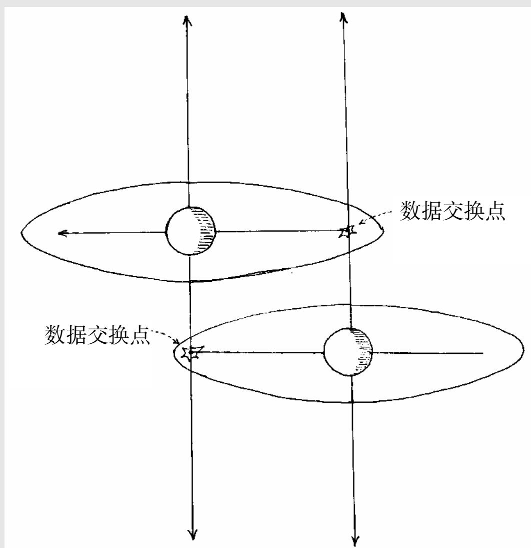
每个原子都依赖于与相邻原子间的力，这就意味着从每个原子向其他所有原子发送信息

中立领域的计算

中立领域（NT）算法计算每个原子与相邻原子间的力，最大距离大约是10埃（1埃等于10^{-10}米）。

假设用A在空间中用x、y和z的坐标表示其位置，记为xA，yA，zA。相应地，B的位置表示为xB、yB和zB。在NT算法中，A通过它的高（zA值）定义的平面发送信息到某个被它定义为“近”的距离。同时，A也通过xA和yA定义的纵轴发送信息，达到同样的最大距离。类似地，B也通过zB值定义的平面和xB、yB定义的纵轴发送信息。当A和B的信息在平面和纵列的交集处相遇——xA、

yA、zB——就能计算出A对B的影响。当A和B的信息相遇在xB、yB和zA，就可计算出B对A的影响。



每个原子都在由其z值定义的平面上和由x、y值定义的轴上，向10埃距离内的其他原子发送信息（z是垂直的，因此平面是水平的，并根据x和y的值延伸）。当两个不同原子的信息相遇时，一个原子对另一个原子的作用力就可以计算出来

中立领域方法能加快传统并行集群计算分子动力学的速度，但肖已经不满足于为他自己的并行机安东（Anton）设计算法了（安东这个名字来自于17世纪显微镜制造者安东尼・范・列文虎克）。安东有512个定制芯片，每个芯片有32个专用的28级运算管道。该运算管道是特别为分子动力学模拟中超高速计算成对粒子间相互作用而设计的，其中依次排列着很多小的运算单元，而且每隔800兆赫的时钟周期，运算管道就能得出结果，这个能力差不多相当于一个传统处理器的50次运算操作。

嵌入安东512个芯片上的管道一起工作时，有效峰值速度超过每秒650万亿次，能执行分子动力学中绝大多数计算密集的部分。额外的计算能力由一批更加灵活

的板载处理元件提供。在正常操作模式下，安东仅靠芯片上的存储即可完成这些计算。一个计算结果通常直接流入另一个处理元件。同“非冯”机一样，安东避免了冯·诺依曼瓶颈。一毫秒的蛋白质动力学模拟结果仅花费数天，而不是10万年。就算是和其他采用类似构造技术的搭载512个处理器的装置相比，安东也要快上一千倍，这个区别来自于设计。肖说：“它并非通用的超级计算机，尤其不擅长执行随机科学程序，但该它做的，它全都做得很好。”

其他并行机能以每天100纳秒的速度模拟大型（24 000个原子）蛋白质–水复合物的运动。而安东每天能模拟差不多10微秒：几乎百倍于其他并行机。出于谦虚的性格，肖回避这种赛马似的比较，只将其归功于专业化。

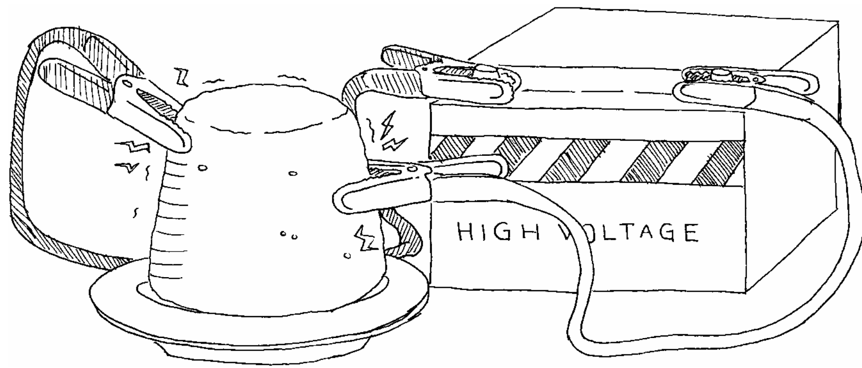
肖的目标应用是计算机辅助药物设计。典型的例子就是去了解抗癌药物如何影响引起癌症的蛋白质。肖和他的团队研究的一种蛋白质突变时会引起某种白血病。这种突变令蛋白质始终处于活化态，导致细胞复制失控。一种较新的抗癌药通过将此种蛋白质锁定在失活态，而延长了很多病人的生命。他说：“我们想知道在原子水平上是什么引起该蛋白质在活化和失活态之间转换，并观察蛋白质运动和形态变化的方式。如果我们能了解这些细节，也许有一天有人能用这些知识来拯救人类。”

就好像历史上安东尼·范·列文虎克用自己强大的新型显微镜首次观察到肌纤维和单细胞组织一样，肖的安东给我们提供了新视角去观察蛋白质的相互作用、现有药物的工作原理以及新药的设计。药物设计是一个长期目标，而肖的短期目标是为科学知识的积累添砖加瓦。他希望自己的实验室是一个小而专的贝尔实验室。他想在由同行评议的期刊上发表他的科学发现，分享新观点，并得到科学界的反馈。

沿着这条路，他的团队将通过一些超长时程的模拟来检验关于分子动力学模

拟准确性的基础工作假设。这些模拟基于原子间作用力的现有模型。一些研究者相信，如果经典的牛顿模型可用于更长时程的分子动力学模拟，那就能重现一大批重要的生物学现象。但没有量子力学效应或其他精妙的物理学理论的支持，可能也就只是可能而已，时程很长的模拟最终也许会步入歧途。计算过程中的一些不精确之处也可能在万亿个步骤后累积起来。

2009年1月26日，肖的团队向科学界公布了安东。安东放置在一间有特殊冷却装置的房间中升起的平台上，体积跟大衣橱差不多。它的512个专用处理器和散热系统的用电量足够30户家庭使用。现场演示播放了一段影片，展示了蛋白质在水中折叠10皮秒（一皮秒等于一万亿分之一秒）的模拟连续帧。折叠过程中，每100皮秒，蛋白质翻转并改变形态。这就有可能跟踪离子对在结构中穿行的轨迹。和肖一起工作的化学家们说他们能在皮秒水平进行任何点和步骤的计算。这曾是分子动力学的未知领域。让我们热烈地欢迎安东这个计算的飞秒镜。





我的思考方式和其他计算机科学家不一样。我的思考是形象化、动觉化的。我有运动梦。我将看到事情发生，并且我自身会成为一个模拟过程。

——乔纳森·米尔斯

第13章

乔纳森 · 米尔斯

自然而然

科学家经常用状态表征变化的微分方程给物理现象建模。举个经济学例子，你银行账户里钱数的变化与你银行账户里的钱数，即目前状态，是成比例的。假设你的银行账户里有100美元，年利率为5%，到年末时会增加5块；如果你有100万，你会看到一年后账户上多了5万。这种钱生钱的关系可以用一个微分方程表达：$dy/dt=ky$，其中k为正。

一说到微分方程，门外汉就望而生畏，但你还是得知道它有两类：常微分方程以及更常见的偏微分方程。常微分方程能描述物理学中的牛顿定律。事实上，为了描述他的定律，牛顿发明了微积分，其中就包括微分方程。偏微分方程可以用于电磁学、热电扩散、量子力学及股票期权的定价等问题的建模。

假设你要解决一个微分方程问题。有一根铁棒，一头在沸水里，一头在冰上，求铁棒上各点的温度。一种方法是用微分方程给这种情况建模，用计算机给方程编程，计算出预测值。这种数字方法遵循以下范式：

问题–建模–编程–计算–得解

另一种方法是建立一个模拟物理系统，为了更快更好，可以选用电系统。这

个模拟系统能测量各处的电荷，推断出铁棒上各个点的温度。这种模拟方法遵循另一种范式：

问题–模拟–测量–得解

如果模拟方法可行，那么就可能比数字方法简单得多、快得多。描述一个自然现象需要微分方程，测量现象告诉我们微分方程如何起作用。假设有两种物理现象X和Y。X的测量值等于微分方程D的计算结果。如果D可以为另一现象Y建模，那么为什么不通过测量X来预测Y？这样不是胜过为D编程计算出Y的预测值吗？模拟方法去掉了中间环节，而且是计算等效的。

模拟的概念早在数字计算机时代之前就出现了。1845年，第一批研究偏微分方程的科学家中有一位名叫古斯塔夫·基尔霍夫，他用一块铜板研究金属中热量的传递。大约100年后，范内瓦·布什用齿轮、曲线绘图器和凸轮组装成“微分分析器”，用以机械化地研究问题。布什预言了互联网最基本的概念：超链接。布什的机器及其电子继承者们都能求解常微分方程问题，甚至比数字计算机尝试这样做还要早。

用电子模拟计算机或是机械模拟计算机编程都必须亲力亲为。科学家要在一块接线盘上各种各样的洞眼里插上电线，把板子搬去实验室，将板子连接到装置上，通上电，等几秒后读出结果。电流通过放大器、电阻和电线节点给出结果。这些装置被称为通用型模拟计算机，能处理算术和常微分方程，但对偏微分方程无能为力。

扩展模拟计算机增加了解偏微分方程的能力。这个想法来自于数学家李·鲁贝尔。因为他对大脑运作方式非常着迷，所以对偏微分方程尤其感兴趣。神经元研究已经表明大脑建立时空模型的方式跟求解偏微分方程很类似。鲁贝尔为他的

机器设计了概念模型，但他觉得根本建造不了。幸好，他是错的。

乔纳森・米尔斯和印地安那大学布卢明顿分校的布莱斯・汉博找到了建造好几代鲁贝尔机的方法。他们造了一系列机器，可以在用硅、泡沫甚至果冻做成的电阻片上计算偏微分方程。最新原型机的程序是这样的：传送电流到一块有25个点的薄板（电源）上的某一个点，从点（电流槽）上带走电流，然后测量电压或者提供一个输出电路。函数根据薄板的结果生成计算的视觉表现，或者将薄板输出与已知答案做比较。然后，米尔斯和汉博仅仅读取电源和电流槽的结果，就能得到更好的答案，或者给另一个薄板提供电流。如果说这种计算方法看起来与众不同，那是因为乔纳森・米尔斯是一位特立独行的计算机科学家。

米尔斯1952年生于密歇根州的卡拉马祖。“别唱那首歌，我能看出来你正想着它。”他说。他指的是格伦・米勒那首著名的“我的卡拉马祖女孩”（其中反复迭唱“zoo，zoo，zoo”）。他的父系血缘来自切罗基族、英国和法国。他的叔叔莱克斯用扒拉出来的零件造了农场里的一切，包括用废车零件和I型梁焊在一起做出拖拉机。米尔斯回忆道：“它看起来很怪，是H形的，上面还装了把摇椅。他骑在摇椅上，但不摇，说是这么骑着更舒服。”

米尔斯的祖父曾是浸礼会牧师，希望自己的儿子能子承父业。但米尔斯的父亲决定去工程公司上班，并且后来当上了史塞克医疗器械公司的质量控制经理，该公司生产外科关节镜等医疗设备。外科关节镜手术是指将小小的、蛇状的关节内窥镜装置插入病人体内，这样医生就可以通过微型摄像机看到感染区域。史塞克公司进入这一领域时拿出来的第一个模型反馈很不好，米尔斯的父亲找到了外科医生用不了这种设备的原因。在一次工程和生产部门的会议上，他解释了原因所在。他拿出两个土豆罐头盒，一个里面装着用来冲洗伤口的生理盐水，另一个是空的，两个罐底各有一枚硬币。老米尔斯请工程师们挑战用关节内窥镜拾取硬

币。他们能搞定空罐子，但无法从装了盐水的罐中取出硬币。原来在设计摄像机时，工程师忘记考虑光线在不同物质中的折射了，他们从来没在溶液中测试过产品。

后来，老米尔斯去吉布森吉他公司，他在那里的工作是维修用来加热吉他的微波烘箱。他还是一位狂热的无线电爱好者，自己设计天线接收无线电信号，其中最远的来自巴塔哥尼亚和西伯利亚。

米尔斯的妈妈在一所为智障儿童开设的学校工作，那里的18岁的孩子只有相当于2岁的智力。为了帮助这些孩子，她写了几百首歌，教他们唱。米尔斯回忆道："妈妈教的歌都是关于生活的，关于他们的生活——他们所看到的世界。"

米尔斯从幼儿园就开始阅读大人的书籍。在20世纪50年代的密歇根，没有为天才儿童开设的学校。米尔斯的妈妈为他申请了跳级测试，结果他一下子就跳到了三年级，比班上同学小一岁，个子也小很多。然而，更大的问题来自老师。三年级有一课是讲火箭的。米尔斯曾经读过一本书《火箭与喷气飞机》（*All about Rockets and Jets*），书里讲述了卡纳维拉尔角的故事、喷气引擎和火箭的发明。老师告诉同学们倒计时从10开始，米尔斯举手纠正老师，说倒计时从100甚至更多开始。米尔斯回忆说："老师眉毛一跳，眼神一凛，说：'你站到前面来，告诉大家是怎么回事。'我一直说一直说，直到最后哭了起来。"

从小到大，米尔斯做了各种各样的装置，他还收集昆虫，学着养蝴蝶。妈妈开车带着他沿着郊区公路跑，收集撞在车上的蝴蝶。在西密歇根大学，他主修拉丁文，能和研究生们一起上小班讨论课。而化学系每个班都有三百人。

钱花完了，米尔斯决定接受美国陆军后备军官训练团的奖学金。当时正是越战期间，米尔斯将被派驻金兰湾。但这时，尼克松总统决定撤军，因此，米尔斯

被派到位于犹他州的达格韦试验场，研究缴获的苏联化工设备。他的老板罗伯特·斯蒂尔曼博士不鼓励他一直在部队工作，而是让他用王安公司的字处理器和能链接到新墨西哥州白沙试验场的DAPRAnet，那里有一台IBM 360。（DAPRAnet是指国防部高级研究计划局计算机网络。）斯蒂尔曼说可以教米尔斯编程。米尔斯回忆说："就像跟小孩说给他糖果。"

米尔斯编程写了个洗车排班表，重新安排了工作流程，将洗车工作由原本需要的1小时缩减为15分钟，不过带来的负面问题是士兵就要连续不停地工作。米尔斯的上级喜欢高效率，但士兵们喜欢洗车1小时的悠闲步调。米尔斯感到了他们的怨气，但从此找到了自己的天职："计算机最接近魔法。编程时，能触摸到不可见的东西，如此强大、可控，又灵活。"

随后，米尔斯去了阿贡国家实验室。阿贡国家实验室建于1946年，是美国第一个国家实验室，位于距芝加哥市区40公里的一个校园内。在那里，米尔斯为逻辑语言Prolog造了一台机器。花了几个月时间，他得到了世界上最快的Prolog编译器。靠着这项工作，摩托罗拉资助了他在亚利桑那州立大学的博士研究工作。他的论文主题就是逻辑编程。

在逻辑编程中，程序员只要描述答案的限制条件，而不是如何得到答案。例如，给定亲子信息，要求找到个体X的所有亲代。在大多数编程语言中，程序都需找到X的父母，然后再找到X父母的父母，以此类推，直到再也找不到父母为止。在Prolog等逻辑语言中，只需要定义好祖先是父母或者另一位祖先的父母即可。米尔斯做的翻译器可以找到全部亲代。

米尔斯发表了他的论文，并决定实现他毕生的梦想——成为一名教授。他得到了印地安那大学布卢明顿分校的教职，开始研究模拟机。回想起来很有意思：他有很强的物理和化学背景，Prolog让他远离了那种按部就班的计算思维，而是

13

更多地从特化解决方案的角度进行计算。

米尔斯觉得自己掌握了技术。他想象把64 000个处理器连在一起。丹尼·海利斯是思维机公司的创始人之一，该公司生产了由神经元启发的一系列计算机。海利斯造了一台机器，试图模拟大脑工作。连接这些处理器需要大量的电线。对数字设计师来说，这需要大量开销。但难不倒米尔斯："我问自己，你怎么才能让电线计算呢？"

疯狂的问题有时候会通向光明大道。在数字计算机上对2个64位数做加法需要上百个晶体管、电阻和一些电线。但现在假设我们将电流X导入电线，电流Y导入另一条电线，两电线与第三条电线接合。那么第三条电线中会有多大电流？你多半已经猜到了：X+Y。就算不懂电学，只要曾经看到过两条河流的汇合，就能明白。支流汇合后，新的河流就有了两条支流的水量。因此，三条电线就能做加法，这可以替代大量电线。

1993年，米尔斯对伊利诺伊大学香槟分校的数学家李·鲁贝尔发表的一些论文产生了兴趣。他先是通过电子邮件联系上了鲁贝尔，然后，"他寄来了一个棕色信封，里面是散发着雪茄气味的论文。"米尔斯回忆道。这些论文说的是李·鲁贝尔对扩展模拟计算机的展望。理想形式下，它能比此前的模拟计算机计算更多的函数——特别是偏微分方程。

模拟机和数字机到底有什么区别呢？说点基础的吧。数字计算机把所有数据分解成被称为比特的微小片段。1比特就是一个0或1，因此，对于π这样的数，数字机只能算出近似值。模拟机用电压水平或者电流存储数据。米尔斯打比方说，数字机就像是用乐高积木搭建一切，而模拟机用的是培乐多泥胶。要用积木搭玩具飞机之类的东西，就需要细致的说明书。用粘土来做则不需要什么说明书，只要回忆飞机的样子，然后作模、雕刻，把粘土塑造成你脑海中或者图片上的样子

就行了。

每种机器都有其最佳用途。数字计算机擅长处理文本、储存通讯录和个人信息以及做账——称作*数据处理*的大量计算。模拟计算机在寻找视觉模式、物理过程建模及机器控制等方面表现最好。这两种计算机的底层电路完全不同。数字计算机里是晶体管、电阻，将电压转换为1和0，上层的程序员和用户并不关心底层的物理现象。而模拟机可以使用底层设备提供范围内全部的值，从而可以用更少的设备进行更多的计算。

数字电路设计师可能会回应说，现在往芯片上塞入更多组件很简单，但很难搞清楚怎么处理它们。在一定程度上，这是真的，但数字计算机提速的最主要障碍是需要从存储器中取出数据，完成一个很简单的操作，比如两数相加，再保存回去（这就是在第12章中讨论过的冯·诺依曼瓶颈）。

而模拟计算机允许一个操作流向另一个操作。更好的是，其底层的物理装置能用来体现与所研究的系统一样的数学模型，因此完全回避了求解多个微分方程的需求，不再需要浪费数十亿次加法和乘法所需的时间。

模拟编程的原理

计算机科学家有一个传统，熟悉了一门语言或者新的计算机时，都会写个程序打出“Hello, world”。米尔斯让他的学生们创造一个H，一开始他们都用数字化思维，觉得应该用点矩阵组成H型。“要用模拟的思维来考虑，问自己能不能创造出山川河谷。如果你把它们看成地形图，就能创造出一个H。”这才是模拟计算机的“Hello, world”。

米尔斯通过在某点设置电源，在另一点设置电流槽，来生成山川河谷。在他的最新结构里有25个可控点。他已经找到了电源和电流槽的模式，可以生成字母表上的全部字母。这些字母每次看起来都完全一样吗？好吧，答案是一样也不一样。他可以生成一个H，何时看起来都像一个H，但并非完全一致。

诚然，做一个简单的字母也需要大量工作。但这种通用的方法可以应用于更难的问题。杜克大学的腓特烈·奈肖特观察到设置了边界和内部条件的偏微分方程，可以生成任意一只蝴蝶翅膀上的全部图案模式。谢尔·桑韦德发现蝶翅模式能组成英文词汇表上的全部字母和数字。米尔斯结合了以上两种观察结果，创造出了蝶翅的通用模式生成器，参看附注栏“蝶翅、小苏打和纸巾”。

蝶翅、小苏打和纸巾

1952年，现代数字计算之父阿兰·图灵用反应扩散方程，首次提出了对生物模式构造（**形态发生**）的深入描述。看名字就知道，**反应扩散方程**适用于研究元素如何与其他实体相互作用以及在其所处环境中传播。

利用求解偏微分方程的算法，数字计算机可以为形态发生建模。这些算法使用一个或者多个有特定边界值的矩阵。在边界条件下操作矩阵，从而对方程系统进行评估。方程的每个部分都与算法的部分保持一致。矩阵翻译成存储器中的元素、将方程翻译为加减乘除等简单的算术操作，这些对数字计算机程序来说都非常容易操作。这些操作的重复序列组成了算法，由遍历矩阵元素的嵌套循环（循环中的循环）所控制。

另一方面，扩展模拟计算机求解偏微分方程和模拟系统行为，靠的并不是算法，而是自然属性。举个简单的例子，我们可以用厨房中随手可得的材料来了解模拟计算机的工作方式。使用以下材料——小苏打、醋、食用色素、纸巾、量杯和平底锅，你就能演示蝶翅模式的形成。

首先，按照平底锅的尺寸，剪好纸巾并把它铺在平底锅里。然后取一量杯用食用色素染红的小苏打溶液、一量杯醋。将小苏打溶液慢慢倒在纸巾上，红色扩散成一个圆形。

因为所有的物理过程都被自然法则统治，所以纸巾和蝴蝶翅膀的工作原理也很相似，只是时间有所不同。在这个“厨房科学”实验里，纸巾染色只需要几秒钟，而蝴蝶则是从蝶蛹时期起，细胞接触到化学信号，翅膀的颜色逐渐形成。

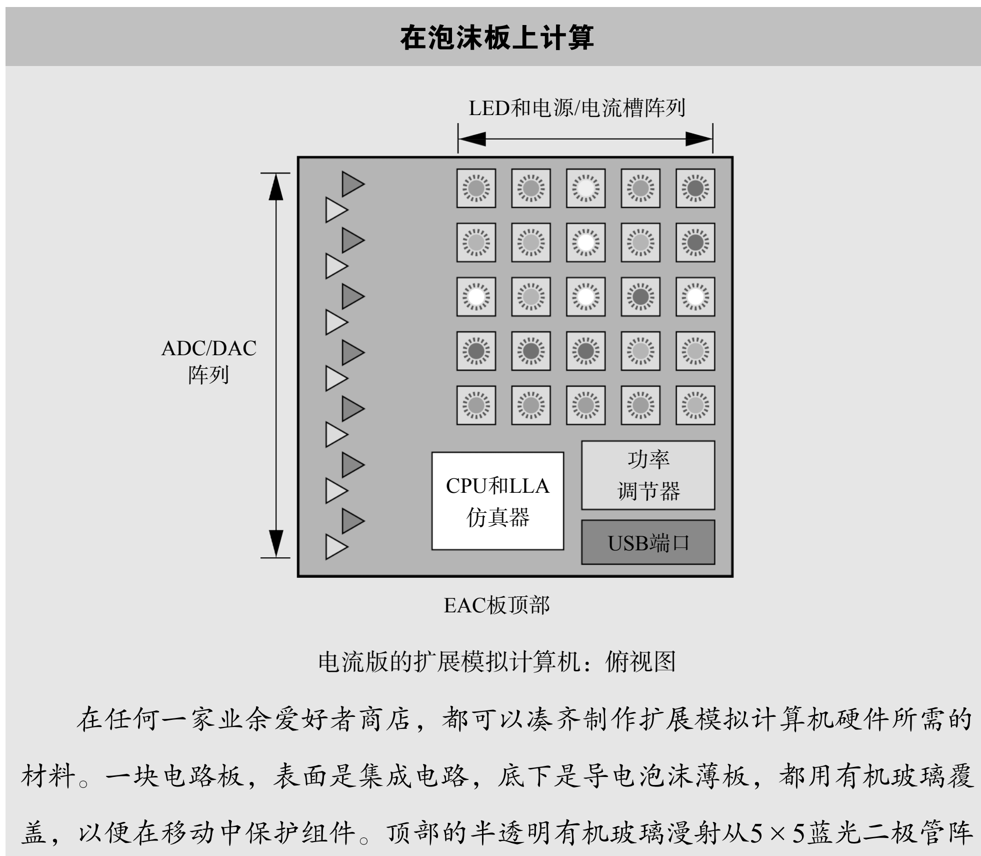
在扩展模拟计算机中，电子在硅晶格的原子中扩散，约需1纳秒。数字计算机可以模拟所有这些系统，但必须是间接的，需要按部就班地依程序来。模拟系统直接用自然法则操作。在扩展模拟计算机中，电源和电流槽可以放置于栅格中25个点中的任何一个上。为了给蝶翅模式的生成建模，电源放置在中心，而电流槽放置在栅格的四边。

提醒一下，如果在纸巾下弄出气泡，厨房实验还可以给反应扩散方程建模。首先，将醋轻轻倒在粉色圆环的中心，仔细观察，就会看到虽然表面张力将纸巾拉平，但会有一个或多个小“气球”或者说是气泡出现在纸巾下。因为醋和小苏打反应，生成二氧化碳，所以圆形气球从中心开始扩大。但由于气体缓慢地从小气球中逃逸，如果够幸运，再等几分钟，你就会看到气球根据纸巾类型形成了由小气泡组成的圆形或者空心菱形。这个现象模拟了图灵所描述的形态发生以及奈肖特所说的蝶翅模式形成。

创造字母H和蝶翅图案建模可能会像机智问答一样让你为之一振，但扩展模拟计算机具有成为超级计算机的潜能，可以迎接重大挑战问题（Grand Challenge problems，指计算机科学家发现的那些非常难以解决，但又对科学和社会非常重要的难题）。

米尔斯的想法是利用扩展模拟计算机提出解决方案，然后由数字协同处理器去验证。如果答案错了，协同处理器就发送反馈信号给模拟处理器，让模拟处理器重新调整电源和电流槽的值，以期改进模拟机的猜测。米尔斯首先将“模拟机提出方案、再通过数字机验证”的策略应用在人工视网膜的制作问题上。他的思路是将图像转化成对可导电泡沫板输入的电信号，然后将泡沫板上产生的一系列电压值与已知物体的电压属性进行对比。计算机一开始假定第一个物体是一类相似物体的范本。当扩展模拟计算机即人工视网膜“看到”一个不属于已知种类的物体时，逻辑门函数就通过数字反馈被修正，并把新物体归到自己的类别中（参见附注栏内容“在泡沫板上计算”）。

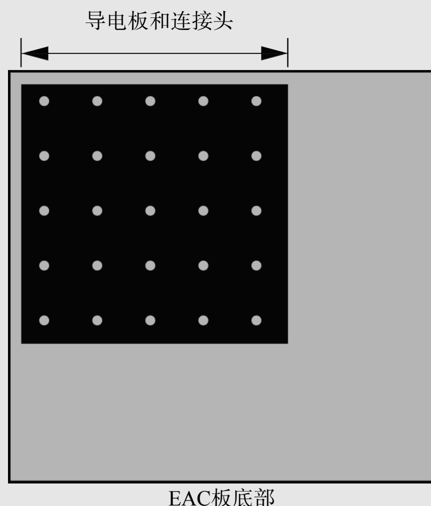
在泡沫板上计算



电流版的扩展模拟计算机：俯视图

在任何一家业余爱好者商店，都可以凑齐制作扩展模拟计算机硬件所需的材料。一块电路板，表面是集成电路，底下是导电泡沫薄板，都用有机玻璃覆盖，以便在移动中保护组件。顶部的半透明有机玻璃漫射从5×5蓝光二极管阵

列发出的光，该阵列粗略地显示了输出的变化，即电压梯度的变化。各自位于半透明有机玻璃边线中间的小孔每四个一组，为一个小针脚提供入口，如上图中的暗点所示。针脚使有机玻璃计算板四条边都能与别的板子相连，这样一堆机器就能建立三维系统模型。米尔斯计划用这种方式配置模拟计算机，解决诸如蛋白质折叠之类的问题。



电流版扩展模拟计算机：仰视图

电路板底部是一块黑色塑料导电泡沫，同样的材料经常用来在运输时保护集成电路。米尔斯开玩笑说有同事买计算机的时候，他用包装材料也能给自己做一台。泡沫板用5×5的金属针脚固定在电路板上，每个针脚都和顶部有特殊用途的电路连接，其中就有发光二极管。每条电路都能独立实现以下三种功能：(1) 给泡沫板提供正/负电流；(2) 直接从泡沫板读取电压；(3) 或者读取电压，并用卢卡西威兹逻辑函数进行测量[①]。

① 卢卡西威兹门以波兰数学家让 · 卢卡西威兹的名字命名。布尔门是典型的数字门，输入和输出都是0和1。与数字门不同，输入卢卡西威兹门的可以是0和1之间连续的值，输出也是如此。举个例子，布尔换流器把输入的1变成输出的0，或者把输入的0变成输出的1。输入卢卡西威兹换流器的值为x（0–1之间），输出则是1–x。连续电路的好处在于能得到一系列无限的结果值。

这个设计看起来简单，但能利用泡沫板的物理特性进行计算。观察泡沫板行为最简单的角度就是用基尔霍夫电流定律求出来自25个针脚的正负电流之和。**基尔霍夫电流定律**的描述很简单：流入节点的电流之和等于流出该节点的电流之和。流经薄板的电流遇到电阻，就会有可测量的电压降。薄板的电容虽然很小，但可以引起延迟，对于计算振荡系统的方程非常有用。下一阶段的建模要利用电子在连续的薄板中扩散这一事实。因为这种扩散可以用偏微分方程建模，程序员能通过测量电压“求解”此类方程。

其他材料也通过电线与输入针脚连接。硅集成电路是产生电压梯度的传感器，这对于电子视网膜来说非常有用。果冻裹住的三维触点阵列可以为三维系统（天气预报、蛋白质折叠等）建模。薄板能让机器使用者直接把自然属性当作最基本的“指令组”。混合匹配不同的材料提高了扩展模拟计算机的计算能力。

连接到数字计算机主机（比如传统个人电脑）的扩展模拟计算机开始运行，主机通过USB端口向扩展模拟计算机发送数据。数字信号转化为模拟信号，使电流通过泡沫板底部5×5针脚阵列中的部分针脚。其他针脚测量电压，并将电压值通过连续的卢卡西威兹逻辑门传回数字主机，主机可以通过标准程序进行操控。

利用反馈信息在日常生活中是如此司空见惯，以至于我们几乎意识不到。眼睛的反馈让我们能安全地在新路上开新车。就算开同一辆车在同一条路上跑了很多年，你也绝不会想蒙上眼睛来一次。狗叼飞盘也是利用了反馈，如此才能在不规则的风里还能咬住飞盘。未来，利用模拟计算的能力和速度提出解决方案，再用数字计算机的精确性（甚至人类的直觉）进行验证，这种方式对我们的孙辈来说可能会很平常，就好像iPhone在儿辈中普及一样。

米尔斯意识到自己的思维方式跟大多数在数字文化中浸淫多年的同事截然不同。他说，一个典型的计算机科学家“擅长控制，他们想提前明确所有东西，并控制过程，以便机器正确运行”。米尔斯通过建模（例如分子球棒模型）研究

问题，想象如何在电路中实现。“我的思考是形象化、动觉化的。我有运动梦。”闲暇时间里，米尔斯会将新世纪音乐和梦幻般的艺术品结合起来欣赏。

思维与众不同有风险。米尔斯申请终身教职的时候，一位同事说他在本领域内非常杰出，还补充说米尔斯是该领域内唯一称得上杰出的人。米尔斯得到了终身教职，但依然有人质疑他的方法是否可以实现。米尔斯非常乐观，认为新一代计算机科学家会赞赏模拟方法的简洁。他相信，从根本上讲，以数学和程序的视角来思考问题比他的方法要困难得多。

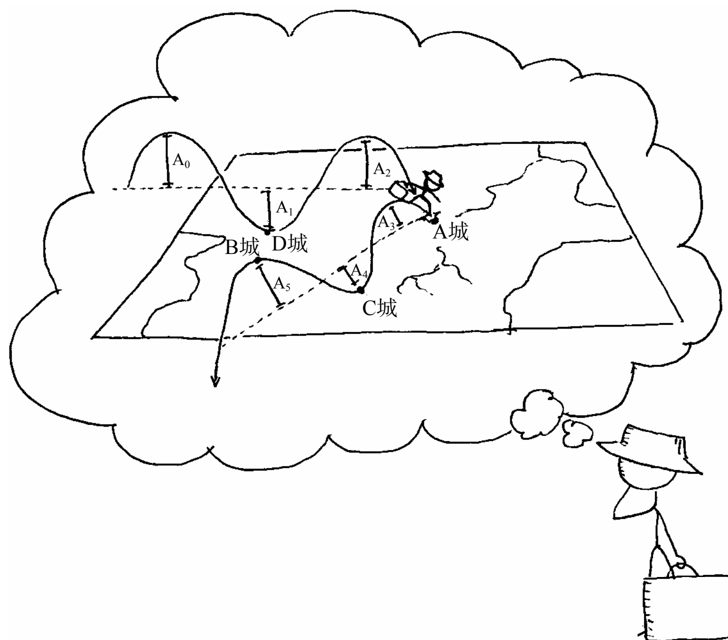
扩展模拟计算机利用结构而不是程序，这让数字计算先驱约翰·巴克斯在1977年提出的一个问题重新回到人们的视线[①]。数字计算机的程序必须很明确。程序可能会引起意料之外的结果。巴克斯称之为“冯·诺依曼式编程”，并询问计算是否可以从中解放出来。米尔斯认为可以，但靠的不是数字计算机。

模拟计算机利用材料的物理相似性进行计算。在扩展模拟计算机中最重要的组件（导电薄板）上，“计算元件”就是原子或分子本身。它的数字组件——二元逻辑门、存储器和电线——只需要做它们擅长的事。在与纽约大学的科学家们关于蛋白质折叠的合作研究中，米尔斯使用的数字组件处理近距相互作用，而模拟组件处理远距相互作用。

在某种程度上，这种分工并不意外。另一位叛逆的科学家理查德·费曼也认为数字计算未必是为自然建模的最佳途径，至少对某些事情来说不是。米尔斯说：“想想费曼的著名问题：为什么即使是一丁点儿的自然建模，数字计算机都要大费周章呢？”

13

① 关于巴克斯的生平，请参考我们的第一本书《奇思妙想：15位计算机天才及其重大发现》。





一个世纪以来，物理学家们成功地让人们都以为量子力学很难。但我发现，一旦抛开物理部分，量子力学只不过是带负号的概率论。这是个启示。我不太懂物理，但我会矩阵和矢量的乘法。

——斯科特·阿伦森

第14章

斯科特·阿伦森

寻找物理新法则

到写作本书的时候为止，还没有人造出拥有大量量子比特的量子计算机。然而，著名的理论学者们已经为量子计算机设计了非凡的算法；其他人则在探索这些主要还仅存在于理论上的机器的理论限制；政府秘密机构也已经投入巨资建造量子计算机。为什么前景尚不明朗的东西会如此受关注？

动机多种多样。间谍们对它感兴趣，是因为非凡的算法意味着相较于常规计算机，量子计算机破解密码的速度也许是指数级的。一些物理学家相信，如果造出了量子计算机，就能展示无数平行世界的存在。数学家相信量子计算为理解被称为计算复杂性的一类问题打开了一扇窗。数学家斯科特·阿伦森就属于最后这一群体，对复杂性理论的研究让他对量子计算机产生了兴趣。在MIT办公室的墙上，阿伦森挂着一张生日时朋友送的海报，叫做“复杂性动物园”，他正致力于驯服其中的居民。

阿伦森1981年生于费城，成长于华盛顿渡河村。独立战争期间，乔治·华盛顿曾在此地领导部队横渡特拉华河，与黑森雇佣军作战。阿伦森的母亲是英语老师，父亲是贝尔实验室的科学作家，当时阿尔诺·彭齐亚斯和罗伯特·威尔逊因发现宇宙微波背景辐射获得诺贝尔奖，老阿伦森负责将这一发现传播给大众。

当其他男孩还在打篮球的时候，小小的阿伦森已经开始思考一些极有科学挑战的问题，比如光速。他回忆说："我对这个基础性限制的理念很感兴趣，还有设计以光速飞行的宇宙飞船。当时我并不明白有质量的物体都无法达到光速。"阿伦森还被他在计算机上创造的虚拟世界迷住了。他常常踱来踱去，想着怎么做出自己的电子游戏。他回忆道："就好像你能创造整个宇宙。我不明白这是怎么发生的。也许有好几千穿着白大褂的人，像装配一架747飞机一样，装配出一个电子游戏。"

有一天，一个朋友把自己Apple II里的一个太空游戏给11岁的阿伦森看。那个朋友有游戏的代码，他解释了一行代码的变动怎样让游戏发生变化。阿伦森说："这就像知道了婴儿从何而来，我既生气又失望，怎么没人早点告诉我。"之后，他更进一步，想的不是编程，而是创造自己的编程语言，以便做出Apple BASIC无法完成的事情。他承认道："当时我还没觉得所有的编程语言都是邱奇图灵论题的残羹冷炙。"

邱奇图灵论题断言，只有整数运算、循环指令序列、条件停止等指令的简单编程语言，其表达能力并不比数字计算机差。任何语言都能把复杂计算表达得更简单，但从根本上来说，这并不能让你用计算机做更多的事情。

接下来的几年里，阿伦森在空余时间研究实验计算机科学。他创造了细胞自动机和分形（在更小尺度上自我复制的几何图案），看着它们到处跑。阿伦森12岁时消磨时光的方式和斯蒂芬·沃尔弗拉姆在2002年出版的*A New Kind of Science*一书中提到的有相似之处。和沃尔弗拉姆不同的是，阿伦森从未做过实验。他说："我不做饭，我不擅长手工。"相反，他关注的是计算机程序的限制，它能做什么，不能做什么。

要找到学校的限制就简单多了。他回忆道："我觉得自己就像进了监狱，等待法官判决。"阿伦森在学校里既无聊又焦躁。当他父亲找到一份美国电话电报公司

亚太分公司的工作，全家搬到香港时，阿伦森对上学的挫败感更深了。有人说，到高中就好了，于是，阿伦森跳了一级，进入了高中。“我听说被关在笼子里的动物，打开笼子时，它会继续在笼子里待一会儿，才会慢慢发现自己可以出去。我也是慢慢发现我应该尽快离开高中。”阿伦森回到美国，又跳了一级。但他不愿意再花一年时间上高中，因为学不到复杂的数学。

阿伦森一共跳了三级。后来，他决定转学，参加位于纽约州北部的克拉克森大学的高中同等学历计划，在那里，高中生可以过渡到大学。高中时错过的事情依然缠扰着阿伦森。在他的网站上，他在一篇题为《蜂巢》（The Beehive）的帖子中细数高中时代。他把自己比作孤独的蜜蜂，幻想着一个不真实的场景，身在其中的他忙于闲言碎语、各种活动和青春期的仪式。

在去克拉克森大学前的那个夏天，阿伦森参加了西雅图的华盛顿大学举办的一个数学夏令营，在那儿他接触到了一些与他的终身事业有关的数学概念。“这对我是个启示”，他说。他听了理查德・卡普关于复杂性理论的讲座。作为图灵奖获得者，卡普推动了计算复杂性领域的发展，尤其是展现了NP完全问题（参见附注栏“NP完全：一大家子的难题”）的普遍性。

NP完全：一大家子的难题

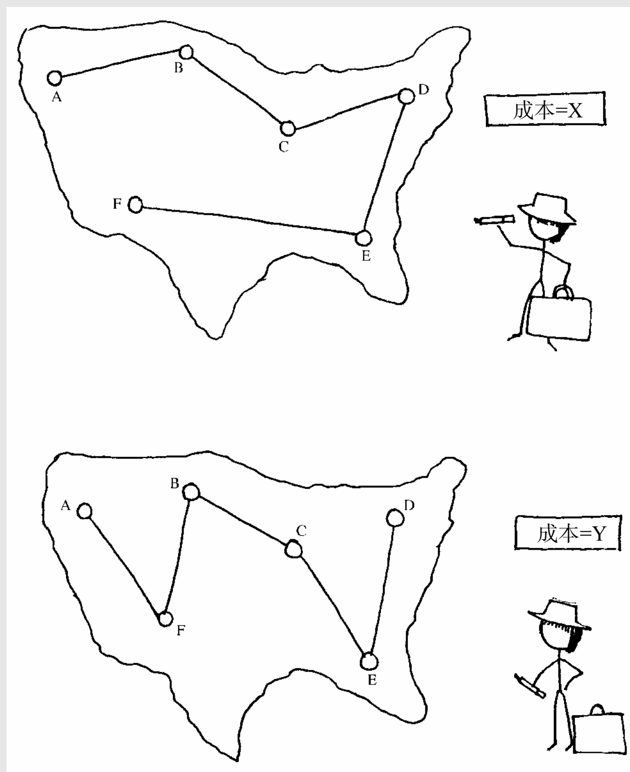
P代表**多项式**（Polynomial）。n的多项式方程有变量和常量，比如，$5n^3+2n-15=0$。NP指的是非确定性多项式。一个计算任务（指“问题”）是P的，即存在一个算法，可以用n的多项式时间解决该问题，问题的“大小”就是n。

举个例子，873 895是否存在于一万个数构成的无序数列中，就是一个大小为10 000（n=10 000）的问题，所需时间与n成正比。其他的问题，例如匹配两条大小为n=10 000的DNA序列，所需时间正比于n^2。所需时间与n^2或者n^{20}成正比的问题都是P的，原因很简单，n（问题的大小）出现在时间表达式的底，而

不是在指数中出现。

如果问题需要穷举所有可能性（俄罗斯数学家生动地称之为perebor，即“暴力”问题），那么所需时间可能是n的指数级（即n可能在特定表达式的指数中）。例如，经常提到的旅行推销员问题，通过尝试所有可能性，找到经过一组城市的最短路线，所需时间大致与n^n成正比。因为指数里有n，所以就是指数级时间。

验证NP问题解集的成本是多项式时间。如果给定一个穿越一系列城市的路线并断言其成本，那么验证该断言所需的时间也与n成正比。实际操作上，如果同一问题的已知解法所需时间都是指数级的，那么这个问题就是**NP完全**的（与P相对）。



旅行推销员问题。推销员有两条路线可选。对给定的任意路线，都能轻松计算出成本。问题就是要找到一条成本低于预算的路线

以指数增长，时间很快就变得不切实际。例如，穿过100个城市的所有路线需要超过十亿个处理器工作十亿秒。因此，对这样的问题来说，实用的策略是使用试探法，快速找到虽然不一定最好，但也非常好的解决方案。在NP完全和P问题之间，横亘着一批没找到多项式时间方法的幽灵问题。这些问题属于P还是NP完全，尚无定论。

幽灵问题中最著名的就是因数分解，这缘于它在密码学中的重要性。你可能还记得在学校里学过的找一个数的素因数。例如，15的素因数是5和3，221的素因数是17和13。找到数n的素因数最笨的方法是试着用所有小于n的平方根的数去除n，看它们是否是n的因数。即使每一次尝试仅需要十亿分之一秒，要找出一个100位数字的所有素因数所需时间都要超过10亿年。这里所需时间是n的位数的指数，而不是n的指数，记为$d(n)$。更快的方法是利用“筛子”大体上降低复杂性，但指数中$d(n)$依旧顽固存在。

拿到高中同等学历时，阿伦森在克里斯 · 林奇的指导下，写了人生第一篇科学论文。林奇是克拉克森大学的计算机科学家，定理机器证明方面的专家。定理机器证明是指用计算机证明数学定理。阿伦森从克拉克森大学毕业后，去了康奈尔大学，与巴特 · 塞尔曼共事，后者是一项机器人足球赛计划的指导老师。阿伦森说:“这让我确信自己不应该当工程师。”让自己的代码和别人的代码协同工作，这不是阿伦森擅长的。

在康奈尔大学暑假的时候，阿伦森在当时被副总统戈尔称为美国科研界“皇冠上的明珠”的贝尔实验室工作。阿伦森为洛夫 · 格罗佛工作了一夏天，格罗佛是量子计算算法的主要发明者之一。虽然还没多少实际建造的量子计算机，但基本的理论模型已经具备了。阿伦森借此机会试图研究出新的计算模型:“水闸已经打开。大家都想知道量子算法能干什么。我怎么可能不感兴趣呢？”

那些志同道合的人都很优秀。物理学家理查德·费曼是最早探索量子计算的人之一。物理学家大卫·多伊奇也是其中之一，而且现在依然活跃在这个领域。在加州大学伯克利分校，阿伦森的博士导师厄麦希·沃兹内尼领导了探索量子计算固有困难的第一次系统研究。

量子力学概要

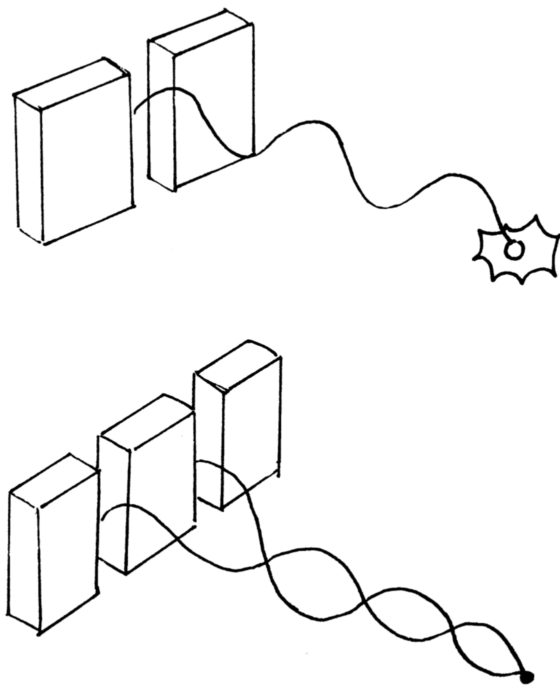
阿伦森采用数学家的方式教授量子力学。20世纪初，物理学课程始于实验观察，其后发展出一系列近似理论来解释观察到的现象，这推动了物理课的发展，最后才接触到核心概念。阿伦森另辟蹊径，告诉自己的学生说："这儿有些叫做振幅的新实体，让我们看看它们会怎么样。"他说："一旦抛开物理部分，量子力学只不过是带负号的概率论。"

让我们试着一步步地理解这个方法。设想掷一对骰子所有互斥的可能结果。每一种可能的概率都在0和1之间，所有概率的总和必为1。量子力学要求我们考虑一种二中择一的概率，称之为振幅。假设振幅取值范围在−1和+1之间，且振幅的所有互斥的可能结果的平方和为1。这样，某种结果振幅的平方就相当于它的概率。

振幅这个概念带来了什么呢？振幅可以叠加，而且既然振幅可以为负（技术上，振幅甚至可以有虚数部分，不过这不是我们要讨论的重点），两个振幅就可以相互抵消——物理学家称之为相消干涉。降噪耳机的工作原理也是相消干涉，只不过是声波而非量子波。相消干涉会导致一些奇怪的效应。

常见的例子是双缝实验。发射单个光子，穿过有两条缝的第一块挡板。在第二块挡板上，你会看到光总不在特定的点上。但若遮住一条缝，光子就能击中这些点。这个结果看起来完全不符合概率论的传统理解。照传统的理解，多开一条

缝只会在第二块挡板上增加可能的目标。



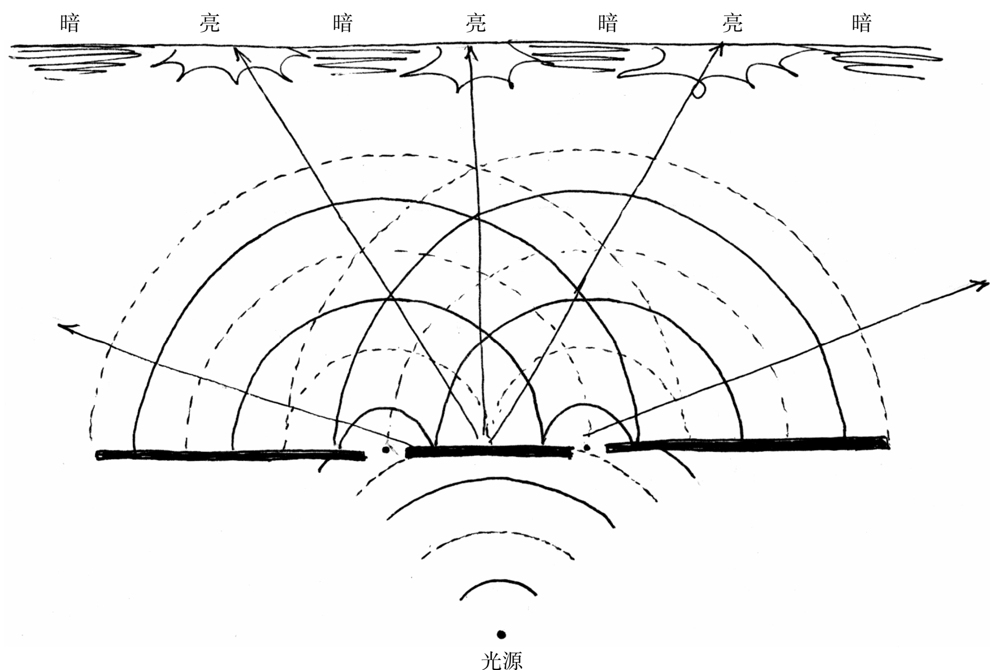
双缝实验。如果只有一条缝（上图），那么虽然密度不同，但各处都会有点光。密度是振幅的平方。双缝实验中（下图），光子的行为是波动的，振幅之间会相互干涉，导致暗点形成

振幅概念给出了一种从数学上预测这种效应的方法。两条缝都打开意味着两个单缝实验振幅的叠加。在特定位置上，波的振幅之和为0，即得到一个暗点。阿伦森说："当人们谈论量子力学所有的奥秘时，其实就是一回事——正负振幅的干涉，只不过出现的形式千奇百怪。"

量子计算

把振幅理解为在−1到+1之间波动的值，有助于理解量子计算机中量子比特，

或量子比特的概念。在经典计算机中，1个比特要么是0要么是1。在量子计算机中，1个量子比特的振幅是0，记为A0，振幅为1的记为A1。如果有两个量子比特，我们就需要把4种可能的振幅情况分别记为A00、A01、A10和A11。对比一下，数字计算机中每个比特的振幅都非0即1，因此两个字节的结合态只有一个非零振幅。如果有三个量子比特，可能的振幅就有A000、A001、A010、A011、A100、A101、A110和A111。



实线表示正振幅，虚线表示负振幅。正负相消处就是暗部

每增加一个量子比特，表示结合态的振幅数量就翻一番。也就是说，*k*个量子比特的结合态有2*k*种振幅。假设某问题有2*k*个可能的值和1个最佳值，我们可以用*k*个量子比特来分配每一种振幅的值。如果能从指数级的可能性里找到最佳值，那么就应该能找到指数加速的圣杯。但主流的科学论文宣传这不过是“瓦罐”，阿伦森说。“事实上，可以认为这就是量子计算中心的瓦罐。”他坚持道。

对某些问题，可以实现指数加速（例如，彼得·肖尔设计的著名算法）。对大多数搜索问题，时间需求大约是按平方根下降的，例如，从一百万步减少到一千步（格罗弗的算法）。尽管格罗弗的方法令人印象深刻，而且很有应用潜力，但并没有实现指数加速。问题在于，对于NP完全问题来说，类似格罗弗的算法就是我们能做到的最好的了。

肖尔的算法把量子算法推上了前台。这一算法是杰出的智慧成就，展示了量子计算机可以更快地进行因数分解。回忆一下，因数分解是找到一个数的素因数。例如，21的素因数是7和3。因数分解的困难构成了密码学的基础，它保障了我们在线支付的安全性。

肖尔的因数分解算法用时大约与被分解数的位数平方成正比。因此，一个1000位的数，“只”需要约一百万步。这比已知最好的经典算法还要快得多，经典算法所需时间约为一千万亿步。我们说“已知”，是因为如果有个密码机构找到了更好的因数分解方法，该机构就能破解很多密码，但他们多半不会告诉任何人。

肖尔将因数分解问题简化为寻找重复序列的“阶”，从而建立了自己的算法：这个问题尤其适合量子计算（参看附注栏“肖尔的算法”）。

肖尔的算法

假设问题是分解一个很大的合数N。合数是素数的乘积，而素数的因数只有1和自己。量子计算本质上是运行一个非常快的子程序。它寻找A关于N的“阶”，其中A是随机选择的与N互质的数。

两个数**互质**，是指它们的最大公约数是1。例如，4和15互质。（如果A和N不互质，则它们有公因数。这个公因数也是N的一个因数，这就给我们的问题提出了一个解决方案，因此，我们先假设自己没这么幸运。）阶是最小的r，使

得A^r mod N=1（mod算符是只保留相除后的余数。例如，13 mod 5等于3，因为13除以5商2余3）。如果r是偶数，则N和（$A^{r/2}+1$）的最大公约数很可能就是N的因数。

让我们看看对于简单的合数15它是如何工作的。取A=4，有4×4 mod 15=1，所以，顺序r为偶数2。因此，（$A^{r/2}+1$）=5，就很可能是15的因子，而且确实是！再举另一个例子：取A=8。也就是暗示r=4，因为84=4096=（273×15）+1，所以84 mod 15=1。因为r为偶数4，计算（$A^{r/2}+1$）=64+1=65。15和65的最大公约数是5，又是15的因数。这种方法可以用于寻找长达百位数的因数。

如果你还想了解更多细节，就去看看阿伦森自己在网站上的描述吧（www.scottaaronson.com/blog/?p）。大卫·多伊奇在*The Fabric of Reality*一书中给出了自然哲学的解释。

肖尔的算法给指数级的突破带来了希望，但它只能解决一个非常特殊的问题。格罗弗的算法虽然提速稍逊，但对一般的黑箱类问题都可用。假设你想知道某个特定的值是否存在于n个值的集合中。若用经典计算机，这个任务大致需要查看集合中的一半元素（n/2）；若用量子计算机，所需步数则为n的平方根。

这种方法又一次利用了振幅的奇特性质。格罗弗描述了一个控制振幅的操作，如果你要找的值在集合中，并且与一个位组合一致，那么对这些振幅实施的特定操作就能放大与该位组合一致的振幅。这个操作就好像是把量子比特的结合态旋转到所需的位组合。经过约n的平方根次旋转，组合就找到了。

查利·本内特、伊桑·伯恩斯坦、吉勒斯·布拉萨德和厄麦希·沃兹内尼证明，每次查询只能做这么多。对这种非结构性问题，n的平方根是最好的结果。如果存在一种针对NP完全问题的量子算法，它必然会利用自身的结构，正如肖尔的算法利用了因数分解问题的结构一样。不幸的是，一般的NP完全问题似乎没有足

够的可利用结构。

阿伦森用地图的例子解释了这个小差错。假设你正用三种颜色给地图上色，相邻国家的颜色不能相同。你并不知道上色的方式是只有一种，还是有上百万种，又或者根本没有。而在因数分解中，你可能不知道因数是什么，但你知道每个正整数都有独一无二的因数分解。因数分解有这种令人欣赏的特性是因为正整数的结构。阿伦森说："这种'底层结构的差异'很重要。我们中很多人都不相信存在一个解决更'普遍'的搜索问题的快速量子算法。"

量子计算机可以指数级提升因数分解的速度，对非结构性问题则是平方根级别的提速。NP完全问题比黑箱问题有更多结构，但不如因数分解。我们能比平方根提速做得更好吗？没人知道，但什么都有可能。阿伦森相信"解决NP完全问题的能力会让我们拥有神一般的力量"。"谈到NP完全问题，我们不仅仅是在谈论航班日程安排（或者就事论事，破解RSA密码系统）。我们说的是自动化直觉或智能。"

"直觉或智能"是指找到我们感官数据的简明表示。解决NP完全问题将带给我们高度精炼的表达。例如，你能找到一个简洁的数学模型去解释股票市场数据、莎士比亚的戏剧或者贝多芬的交响曲，只要这样的模型存在就能被找到。阿伦森说："我们在讨论天文数字的可能性，如果确实存在一种解决方案，那么就高效地找出来——也就是我们说的，NP完全问题是否属于P。"

因此，量子计算机上对NP完全问题的指数加速将有重大意义。阿伦森认为，虽然尚未得见，但物理法则的些许改变就能让它成为现实。1998年，丹尼尔・艾布拉姆斯和赛斯・劳埃德展示了量子态"时间演化的弱非线性"，即改变我们的时空连续性概念，也许可以引领我们设计出在多项式时间内解决NP完全问题的量子计算机。换句话说，物理法则的改变将赋予量子计算机神一般的力量。

阿伦森研究了很多利用物理机制解决NP完全问题的提案，从肥皂泡到闭合时间曲线。在一篇题为“NP完全问题与物理现实”的论文中，他调查了一些重要的提案，甚至自己做了一些实验。阿伦森综述中引用的一篇论文提到，可以用肥皂泡作为模拟计算机来解决NP完全问题（斯坦纳树问题），因为肥皂泡倾向于变形为能量最低的形状。阿伦森做了一个肥皂泡机器，但仅仅5个节点，它就找不到最优解了——肥皂泡对解决旅行推销员问题毫无帮助。用诸如肥皂泡、DNA这样的物理实体进行计算无法解决大规模NP完全问题。理论家阿伦森知道，关于NP完全问题有很多非常复杂的文献，其中有些结果提供的证据指向了——尽管未能证明——对数字计算机来说P和NP确实是不同的。他也知道有证据指出，针对多种问题，在利用广为接受的物理法则的量子计算机上，最好的方案是格罗弗的二次项加速。最后，还有些奇怪的结果显示，如果物理法则改变了，量子计算机就能在多项式时间内解决NP完全问题。

即使在量子计算机而言，目前仍未证明，NP完全问题从根本上比P问题更困难，因此阿伦森决定用物理学家的方式思考。提出对一条物理法则地位的断言，其标准比建立一个数学定理真相要低。首先，不能证明熵永远不会减少，或没有质量为正的物体可以比光速更快。这些“法则”都和观察到的现象一致，而且在假定它们为真的情况下，也存在可以给出正确预测的数学理论，但这并不是证明。NP完全问题的固有困难是否具有跟无法证明的物理法则相同的性质呢？

阿伦森指出，大多数最深层的物理学原理都是不可能性声明。例如，不存在永动机。他说：“让我感到迷惑的是这些原理和已知反例之间的双向关系。”一方面，每次试图推翻物理法则的失败都让它更加可靠。另一方面，有时候物理法则能带来新发现。

雅各布·贝肯斯坦发现了黑洞熵——该理论根据热力学第二定律指出，黑洞

会如宇宙的其余部分一样变得更加无序。由于对熵减少的不可能性的重视，他得到了这个理论。因此，NP困难猜想——或者宽泛一些说，就是NP完全问题在物理世界的顽固性——最终是否应该被看作一个物理原理呢？

阿伦森认为这个答案应该依赖于以下两点：(1) 猜想是否有合适的证据；(2) 接受猜想是否会给新物理理论增加有趣的约束。阿伦森认为，由于基本的物理学原因，解决NP完全问题的勇敢尝试已经失败，正如无法获得加速到与光速相当的相对论速度所需的能量一样。结果是，诸多攻克困难猜想的努力尝试都无功而返。

关于NP完全问题十分顽固（即需要指数时间来解决）的猜想，得到了一些对自然的预测。例如，如果回到过去以影响现在的时间旅行是可能的，量子计算机就能在多项式时间内解决NP完全问题。如果我们设想NP完全问题是顽固的，那么像电影《回到未来》中那样的时间旅行就不可能发生。从对NP完全问题的猜想中得到的物理预测之间相互角力，似乎很奇怪。但自牛顿以来，数学理论已经预测了很多物理结果。这个猜想能帮我们找到万物之理吗？阿伦森有解决这一问题的勇气和天赋，但目前，他什么都没说。

后　　记

1959年，物理学家理查德·费曼[1]在美国物理学会做了题为“底层大有可为：请进入物理新领域”（There's Plenty of Room at the Bottom: An Invitation to Enter a New Field of Physics）的演讲。当时，现代生物学还处在石器时代。DNA在此前数年刚刚被描述。RNA和氨基酸之间的关系还是未知。然而，费曼在演讲中还是谈到了一些可能性：

> 生物学并非简单地写下信息，而是还可以操作信息。生物系统可以非常小。细胞虽小，却非常活跃，它们生产多种物质；它们到处走动，摆动；它们能做各种不可思议的事情：全部都在极小的尺度上完成。而且，它们可以存储信息。想想这种可能性吧，我们也能制造非常小的东西去完成交办的任务——我们能生产在这个层面上操作的物体。

费曼用寥寥数语指出细胞可以计算。按照费曼的说法，其他大有可为的底层还包括直径仅10个原子大小的电线，能控制单个原子的制造能力：两者近期都已经成为可能。在纳米水平上，费曼预期量子效应将大行其道，前景诱人。

① 本书介绍的科学家中有两位获得过纳米技术费曼奖，分别是内德·希曼和保罗·罗斯蒙德。

我们能想象当时的听众中有些人对这个前景嗤之以鼻。但这却是正在发生的事情。问题在于将向何方发展呢？让我们从技术开始谈起吧。

计算将摆脱自我强加的数字电子牢笼。20世纪中期，工业设计学校认识到“功能决定形式”的原则。未来计算可能会发掘一个等效的原则：自然决定形式。如果想要一种可以修复皮肤、骨骼或血管的设备，那么用生命物质（DNA、病毒或者细胞）来制造就比用电子元件合情合理得多。如果想理解量子水平的行为，那么用量子计算机就比用数字计算机更有道理。与之相似的是，如果有液体、材料或者物理作用力方面的问题，那么就应该用模拟物理系统来“测量”，而不是计算微分方程的结果。

计算机将会照顾自己。在细胞或DNA水平的计算必须很健壮，能容忍数千个细胞的死亡和意外。人类社会已经习得了这种健壮性。夏尔·戴高乐用他标志性的讥讽将其概括为：“全世界的墓地都满是不可或缺的人。”

当宇宙飞船在距地球数光日之遥旅行时，它们的计算机必须自己生存下来，毕竟在这么遥远的距离下，地球能提供的帮助十分有限。甚至陆地贸易体系也不得不在几乎没有人为干预下的混沌市场的混战中生存。结果将带来非常聪明的新概念：生存、适应、好好干。

协作提高效益。想像一下这样的未来：监测动脉的细胞计算机探测到你心跳的微小异常，就与具象化你的心脏模型的体外模拟计算机通信，决定动脉是否需要清理。数字计算机接收输出后，订购两周用的特种清理细菌胶囊送到家里。细胞计算机监控进度，判断你是否如预期般康复。单个计算设备不能挽救你，但它们合在一起就可以。

目前为止，我们都像技术专家一样说话。现在，让我们像普通人一样说话。

用于治疗的细胞很容易变成间谍。高速计算机能制造更好的武器。新技术有些吓人吗？我们觉得不吓人。别忘了区区数百枚20世纪50年代的氢弹就足以杀掉绝大多数人类。用于监视的铅笔大小的摄像机已经存在很长时间了，更不用提在标记兴奋时间的起搏器了。本书中描述的计算和自然碰撞的技术，并不足以用来作恶，而是给好事带来了巨大的机遇。

如果想要成为星际公民，我们就需要帮助。复杂的宇宙飞船有成千上万个计算机，还不能像我们的个人计算机那样不可靠。类似地，如果想要延长寿命，我们就需要能适应我们的需求的疗法。最后，如果人类这个物种要存活，我们就不得不确保大规模的工程制品，如核电站、战略导弹等，不会摧毁我们自身和地球。在未来世界，聪明意味着"适应和好好干"，聪明的机器将是我们的朋友、假肢和探险伙伴。

自然计算时间线

1673年，戈特弗里德·威廉·莱布尼茨发明了乘法机。

1805年，约瑟夫–玛丽·雅卡尔基于穿孔卡片制作了织布机。

1821年，查尔斯·巴贝奇设计了用于计算的分析机。

1859年，查尔斯·达尔文在《物种起源》一书中发表了自然选择学说。

1864年，赫伯特·斯宾塞出版了《生物学原理》一书，将进化的关键概念应用于社会科学中。

1866年，格雷戈·门德尔发表的论文“植物杂交实验”成为现代遗传学的基石。

1921年，捷克剧作家卡雷尔·恰佩克在剧本《罗素姆万能机器人》中创造了机器人一词。

1922年，尼尔斯·玻尔因对量子物理的研究获得诺贝尔奖。

1927年，范内瓦·布什和他的MIT团队开始设计微分分析机，一种复杂的、用于解决常微分方程的机械模拟计算机。

1931年，库尔特·哥德尔证明数学中存在虽然为真但无法证明的定理。

1932年，阿隆佐·邱奇发明了λ演算，推进了对编程语言的定义。

1935年，阿兰·图灵定义了一个计算模型，并指出有些问题是计算机无法解决的。

1941年，卡尔·楚泽在德国造出了机电计算机，能存储它自己的程序。

1942年，世界上第一个受控核裂变链式反应堆在芝加哥大学斯塔格运动场被创造出来。

1945年，作为ENIAC（Electronic Numerical Integrator And Calculator，电子数值积分计算机）计划的一部分，宾夕法尼亚大学莫尔学院的约翰·普莱斯波·埃克特和约翰·威廉姆斯·莫奇莱提出了在同一存储空间内存储程序和数据的概念。约翰·冯·诺依曼作为顾问指导EDVAC计划（EDVAC，Electronic Discrete variable Automatic Computer，离散变量自动电子计算机）。他在报告中描述了这一架构，被后人称为冯·诺依曼架构。

1947年，约翰·冯·诺依曼开发了细胞自动机，一种自我复制机的模型。

1948年，自我复制机的论文在加州理工学院举办的希克松研讨会上发表。

1948年，诺伯特·维纳（Norber Wiener）出版了系统论的奠基性著作《控制论：或关于在动物和机器中控制和通信的科学》。

1953年，詹姆斯·沃森和弗朗西斯·克里克用罗莎琳德·富兰克林拍摄的衍射影像发现了DNA双螺旋结构。

1954年，尼尔斯·巴里切利演示了用计算机模拟进化。

1955年，乔纳斯·沙克开始用灭活脊髓灰质炎病毒进行免疫接种。

1956年，约翰·麦卡锡在达特茅斯会议上提出了人工智能这一概念。

1957年，苏联发射了第一颗人造地球轨道卫星斯普特尼克（Sputnik）。

1958年，为应对苏联的导弹威胁，加拿大和美国成立了北美防空联合司令部（NORAD）。

1959年，理查德·费曼在美国物理学会年会上发表了题为“底层大有可为：请进入物理新领域”的演讲，指出了现代纳米科技的挑战。

1962年，工业机器人在通用汽车的装配线上初次登场。

1962年，阿尔伯特·沙宾的减毒脊髓灰质炎口服疫苗开始普遍使用。

1965年，英特尔公司联合创始人戈登·摩尔发表论文称，根据他的观察，集成电路上可容纳的晶体管数量成指数增长，每两年翻一番。

1967年，第一台大规模并行计算机ILLIAC IV在伊利诺伊大学建成。

1975年，约翰·霍兰德出版《自然与人工系统中的适应》，综述了现代基因算法的原理。

1976年，第一个火星着陆器海盗1号传回了这个红色行星表面的照片。

1983年，内德·西曼得到了稳定的非线性DNA结构。

1986年，本田公司创造了自动行走机器人Asimo，以科幻大师艾萨克·阿西莫夫的名字命名。

1993年，伊利诺伊大学的李·鲁贝尔发表了关于扩展模拟计算机的论文，突破了香农的通用模拟计算机的限制。

1993年，莱纳德·埃德曼利用DNA给出了一个类似旅行推销员问题的最优解。

1994年，彼得·肖尔表明，原则上量子计算机在因数分解上比数字计算机快指数倍，为破解一整类密码指明了道路。

1996年，哈尔·埃布尔森，盖瑞·苏斯曼和汤姆·奈特创造了无定形计算，可以只用局部相互作用管理大部分异步并行处理器。

1996年，洛夫·格罗弗指出，原则上量子计算机能用n的平方根时间找到n的匹配项，而数字计算机做不到。

1996年，火星探路者发射成功。

1997年，探路者携带的两个探测机器人成了火星上第一批地质学家。

2004年，IBM蓝色基因成为世界上最快的计算机。

2009年，大卫·肖公开了机器人安东（Anton），它有512个处理器，可以模拟分子动力学。

人名索引

A

阿德里安・斯托伊卡（Adrian Stoica）, 3, 16, 26
阿尔伯特・沙宾（Albert Sabin）, 100, 178
阿莱士・里奇（Alex Rich）, 74
阿兰・图灵（Alan Turning）, 154, 177
阿隆佐・邱奇（Alonzo Church）, 107, 177
阿姆鲁特・巴哈拉勃（Amrut Bharambe）, 4
阿穆特・巴拉姆比（Amrut Bharambe）, 43, 47
阿瑟・惠特尼（Arthur Whitney）, 46
埃卡德・维默尔（Eckard Wimmer）, 99
埃里克・温弗里（Erik Winfree）, 78, 87
埃罗尔・莫里斯（Erroll Morris）, 13
艾弗里・费希尔（Avery Fisher）, 96
艾琳・赖特（Irene Wright）, 83
艾伦・贝尔（Alan Bell）, 108, 117
艾伦・贝尔鲍姆（Alan Berebaum）, 117
艾萨克・阿西莫夫（Isaac Asimov）, 28, 179
艾兹格・迪杰斯特拉（Edsger Dijkstra）, 84
爱德华・弗雷德金（Edward Fredkin）, 109
爱德华・萨皮尔（Edward Sapir）, 135
爱德华・威尔逊（E.O. Wilson）, 121
安东（Anton）, 124, 137, 143

B

巴特・塞尔曼（Bart Selman）, 165
芭芭拉・史翠珊（Barbra Streisand）, 106
保罗・罗特蒙德（Paul Rothemund）, 70, 78, 82, 83
本杰明・沃夫（Benjamin Whorf）, 135
彼得・劳伦斯（Peter Lawrence）, 118
彼得・肖尔（Peter Shor）, 169, 179
伯尼・布尔克（Bernie Yurke）, 90
博比・菲舍尔（Bobby Fischer）, 105
布莱恩・拉扎尔（Brian Lazara）, 23
布莱斯・汉博（Bryce Himebaugh）, 149
布鲁斯・鲁宾逊（Bruce Robinson）, 76, 80

C

查尔斯・巴贝奇（Dave Cummings）, 177
查尔斯・达尔文（Charles Darwin）, 177
查利・本内特（Charlie Bennett）, 85, 170

D

大卫・多伊奇（David Deutsch）, 166, 170
大卫・鲁梅尔哈特（David Rumelhart）, 138
大卫・斯科特（David Scott）, 14
大卫・肖（David Shaw）, 136, 137, 179
大卫・雅各布森（David Jacobsen）, 47, 48
戴夫・卡明斯（Dave Cummings）, 23
戴夫・米勒（Dave Miller）, 14
戴夫・史麦丝（Dave Smyth）, 23
丹尼尔・艾布拉姆斯（Daniel S. Abrams）, 171
丹尼尔・库尔（Daniel Coore）, 110, 117, 119

E

厄尔・亨特（Earl Hunt）, 57
厄麦希・沃兹内尼（Umesh Vazirani）, 166, 170

F

范内瓦・布什（Vannevar Bush）, 148, 177
腓特烈・奈肖特（H.Fredrik Nijhout）, 154
弗兰克・盖里（Frank Gehry）, 107
弗朗西斯・克里克（Francis Crick）, 98, 178

G

盖瑞・苏斯曼（Gerry Sussman）, 116, 117, 179
盖西・塔库提（Gaisi Takeuti）, 128
盖伊・斯蒂尔（Guy Steele）, 107
戈登・摩尔（Gordon E. Moore）, 123, 179
戈特弗里德・威廉・莱布尼茨（Gottfried Wilhelm Leibniz）, 177
格雷格・韦尔茨（Greg Welz）, 23,
格雷戈・门德尔（Gregor Mendal）, 177
格里戈・孟德尔（Gregor Mendel）, 98
格林内尔・穆尔（Grinnell Moore）, 12
格伦・里夫斯（Glenn Reeves）, 3, 16, 17

H

哈尔・埃布尔森（Hal Abelson）, 116, 117, 179
汉斯・莫拉韦克（Hans Moravec）, 9
赫伯特・斯宾塞（Herbert Spencer）, 177
霍华德・艾肯（Howard Aiken）, 45
霍华德・加德纳（Howard Gardner）, 7

J

吉勒斯・布拉萨德（Gilles Brassard）, 170
简・斯奈普斯赫特（Jan L. A. Van de Snepscheut）, 84
杰克・霍洛韦（Jack Holloway）, 108
杰克・洛夫莱斯（Jake Loveless）, 4, 42, 43, 135
杰拉尔德・苏斯曼（Julie Sussman）, 104, 105
金姆・葛斯特罗（Kim Gostelow）, 23

K

卡尔・埃维特（Carl Hewitt）, 107
卡尔・楚泽（Karl Zuse）, 177
卡尔・施耐德（Karl Schneider）, 23
卡雷尔・恰佩克（Karel Čapek）, 177
凯西・麦克多诺（Kathy McDonough）, 77
科林・安格尔（Colin Angle）, 12, 13, 14
克劳德・香农（Claude Shannon）, 106
克里斯・林奇（Chris Lynch）, 165
肯・艾弗森（Ken Iverson）, 45
库尔特・哥德尔（Kurt Gödel）, 177

L

拉蒂卡・纳格帕（Radhika Nagpal）, 71, 110, 114
拉格纳萨・拉杰库马尔（Ragunathan Rajkumar）, 24
拉吉夫・德赛（Rajiv Desau）, 14
莱纳德・埃德曼（Leonard Adleman）, 179
莱恩・阿德尔曼（Leonard Adleman）, 78
雷・麦克莱伦（Rae McLellan）, 117
李・鲁贝尔（Lee Rubel）, 124, 148, 152
理查德・费曼（Richard Feynman）, 84, 159, 166
理查德・卡普（Richard Karp）, 163
林恩・康韦（Lynn Conway）, 108
卢莎（Lui Sha）, 24
路易斯・奎尔斯（Louis Qualls）, 3, 32, 33
路易斯・沃尔珀特（Lewis Wolpert）, 118
伦纳德・阿德尔曼（Len Adleman）, 85, 87
伦纳德・莱尔曼（Leonard Lerman）, 76
罗伯特・安德森（Robert Anderson）, 19
罗伯特・富尔（Robert Full）, 14
罗伯特・斯蒂尔曼（Robert Stearman）, 151
罗德尼・布鲁克斯（Rodney Brooks）, 3, 6, 7
罗恩・韦斯（Ron Weiss）, 110
罗莎琳德・富兰克林（Rosalind Franklin）, 73, 98, 178
洛夫・格罗佛（Lov Grover）, 165, 179

M

马文·明斯基（Marvin Minsky），106, 109
迈克·德里曼（Mike Deliman），23
蒙蒂·代诺（Monty Denneau），124, 126, 127
莫·霍华德（Moe Howard），105

N

南希·莱韦森（Nancy Leveson），4, 54, 55
内德·西曼（Ned Seeman），69, 72, 78
尼尔·戴蒙德（Neil Diamond），106
尼尔斯·奥尔·巴里切利（Nils Aall Barricelli），2
尼尔斯·玻尔（Niels Bohr），177
诺尔曼·马戈利斯（Norman Margolis），109
诺伯特·维纳（Norbert Wiener），178

P

帕姆·吉冈（Pam Yoshioka），23
皮埃尔·居里（Pierre Curie），73
皮特·帕斯夸（Pete Pasqua），34

Q

乔纳森·凯利（Jonathan Kelley），88
乔纳森·米尔斯（Jonathan Mills），124, 146, 147, 149
乔治·肖伯纳（George Bernard Shaw），137

R

瑞克·阿卡兹（Rick Achatz），23

S

赛斯·劳埃德（Seth Lloyd），171
史蒂夫·施托尔珀（Steve Stolper），23
史蒂夫·斯凯纳（Steve Skiena），70, 94, 95
史坦尼斯劳·莱姆（Stanislav Lem），28
斯蒂芬·克莱尼（Stephen Kleene），107
斯蒂芬·沃尔弗拉姆（Stephen Wolfram），109, 162
斯科特·阿伦森（Scott Aaronson），125, 160, 161
斯坦尼斯拉夫·乌拉姆（Stanislaw Ulam），109

T

汤姆·奈特（Tom Knight），111
唐·迈耶（Don Meyer），23
唐娜·雪莉（Donna Shirley），13
藤田文章（Humiaki Huzita），119
托马斯·亨特·摩尔根（Thomas Hunt Morgan），98
托马索·托福利（Tommaso Toffoli），109

X

夏尔·戴高乐（Charles de Gaulle），175
西摩·克雷（Seymour Cray），138

Y

雅各布·贝肯斯坦（Jacob Bekenstein），172
雅克·库斯托（Jacques Cousteau），84
伊莱·惠特尼（Eli Whitney），1
伊桑·伯恩斯坦（Ethan Bernstein），170
英戈·雷兴贝格（Ingo Rechenberg），2
约翰·P. 莱霍茨基（John P. Lehoczky），24
约翰·冯·诺依曼（John von Neumann），123, 178
约翰·普莱斯波·埃克特（J. Presper Eckert），178
约翰·霍兰德（John Holland），2, 30, 179
约翰·科扎（John Koza），30
约翰·麦卡锡（John McCarthy），106, 107, 178
约翰内斯·谷登堡（Johannes Gutenberg），1
约纳斯·沙克（Jonas Salk），70, 100

Z

詹姆斯·沃森（James Watson），98, 178
朱莉·苏斯曼（Julie Sussman），116

参考文献

第 1 章　罗德尼·布鲁克斯

Brooks, Rodney A. “Elephants Don’t Play Chess.” *Robots and Autonomous Systems* 6 (1990): 3–15.

Brooks, Rodney A., and Anita M. Flynn. “Fast, Cheap and Out of Control: A Robot Invasion of the Solar System.” *Journal of the British Interplanetary Society*, 1989, 478–85.

Spenko, M. J., G. C. Haynes, J. A. Saunders, A. A. Rizzi, R. J. Full, and D. E. Koditschek. “Biologically Inspired Climbing with a Hexapedal Robot.” *Journal of Field Robotics*, April 2008, 223–42.

第 2 章　格伦·里弗斯和艾德里安·斯托伊卡

Bell, Jim. Postcards from Mars: The First Photographer on the Red Planet. NewYork: Dutton, 2006.

Koza, John R., F. H. Bennett III, D. Andre, and M. A. Keane. *Genetic Programming III—Darwinian Invention and Problem Solving.* San Francisco: Morgan Kaufman, 1999.

Reeves, Glenn. “What Really Happened on Mars?” *Risks Digest* 19, no. 49 (1997).

Reeves, G. E., and J. F. Snyder. “An Overview of the Mars Exploration Rovers’ Flight Software.” Paper presented at the IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI, October 10–12, 2005.

Stoica, Adrian, Ricardo Zebulum, Didier Keymeulen, Rajeshuni Ramesham, Joseph Neff, and Srinivas Katkoori. “Temperature-Adaptive Circuits on Reconfigurable Analog Arrays.” Paper presented at the fi rst NASA/ESA Conference on Adaptive Hardware and Systems, Istanbul, Turkey, June 15–18, 2006.

Stoica, Adrian, Ricardo Zebulum, Didier Keymeulen, Raoul Tawel, Taher Daud, and Anil Thakoor. “Reconfi gurable VLSI Architectures for Evolvable Hardware: From Experimental Field Programmable Transistor Arrays to Evolution-Oriented Chips.” *IEEE Transactions on Very Large Scale Integration Systems* 9, no. 1 (2001): 227–32.

第 3 章 路易斯・奎尔斯

Mason, Lee, David Poston, and Louis Qualls. “System Concepts for Affordable Fission Surface Power.” Paper presented at the Space Technology and Applications International Forum (STAIF–2008), sponsored by the Institute for Space and Nuclear Power Studies at the University of New Mexico, Albuquerque, NM, February 10–14, 2008.

第 5 章 南希・莱韦森

Leveson, Nancy. System Safety Engineering: Back to the Future (a work in progress available on Leveson’s website, http://sunnyday.mit.edu/book2.html).

第 6 章 内德・希曼

Seeman, Nadrian C. “Nucleic-Acid Junctions and Lattices.” *Journal of Theoretical Biology* 99, no. 2 (1982): 237–47.

Zhang, Y., and Nadrian C. Seeman. “The Construction of a DNA Truncated Octahedron.” *Journal of the American Chemical Society* 116 (1994): 1661–69.

第 7 章 保罗・罗森蒙德

Adleman, Leonard M. “Computing with DNA.” *Scientifi c American*, August 1998, 54–61.

Adleman, Leonard M. “Molecular Computation of Solutions to Combinatorial Problems.” *Science* 266 (1994): 1021–24.

Amos, Martyn. Genesis Machines: The New Science of Biocomputing. New York: Overlook Press, 2008.

Benenson, Y., B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro. “An Autonomous Molecular Computer for Logical Control of Gene Expression.” *Nature* 429 (2004): 423–29.

Charles H. Bennett. “The Thermodynamics of Computation—A Review.” *International Journal of*

Theoretical Physics 21 (1982): 905–40.

Rothemund, Paul W. K. “A DNA and Restriction Enzyme Implementation of Turing Machines.” In *DNA Based Computers: Proceedings of a DIMACS Workshop, April 4, 1995, Princeton University* (DIMACS Series in Discrete Mathematics and Theoretical Computer Science 27), edited by Richard J. Lipton and E. B. Baum, 75–119. Providence, RI: American Mathematical Society, 1996.

Winfree, Erik. “On the Computational Power of DNA Annealing and Ligation.” In *DNA Based Computers: Proceedings of a DIMACS Workshop, April 4, 1995, Princeton University* (DIMACS Series in Discrete Mathematics and Theoretical Computer Science 27), edited by Richard J. Lipton and E. B. Baum, 199–221. Providence, RI: American Mathematical Society, 1996.

Winfree, Erik, Furong Liu, Lisa A. Wenzler, and Nadrian C. Seeman. “Design and Self-Assembly of Two-Dimensional DNA Crystals.” *Nature* 394 (1998): 539–44.

Yurke, Bernard, Andrew J. Turberfi eld, Allen P. Mills Jr., Friedrich C. Simmel, and Jennifer L. Neumann. “A DNA-Fuelled Molecular Machine Made of DNA.” *Nature* 406 (2000): 605–8.

第 8 章　斯蒂夫 · 斯凯纳

Coleman, J. R., D. Papamichail, S. Skiena, B. Futcher, E. Wimmer, and S. Mueller. “Virus Attenuation by Genome-Scale Changes in Codon Pair Bias.” *Science* 320 (2008): 1784–87.

Skiena, Steve. *Calculated Bets: Computers, Gambling, and Mathematical Modeling to Win*. Cambridge: Cambridge University Press, 2001.

第 9 章　杰拉尔德 · 苏斯曼

Abelson, Harold, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F. Knight Jr., Radhika Nagpal, Eric Rauch, Gerald Jay Sussman, and Ron Weiss. “Amorphous Computing.” *Communications of the ACM*, May 2000, 74–82.

Weiss, Ron, Thomas F. Knight Jr., and Gerald Sussman. “Genetic Process Engineering.” Chapter 4 in *Cellular Computing*, edited by Martyn Amos. Oxford: Oxford University Press, 2004.

第 10 章　拉迪卡 · 纳格帕

Coore, Daniel, Radhika Nagpal, and Ron Weiss. “Paradigms for Structure in an Amorphous Computer.” A. I. Memo no. 1614. Cambridge, MA: MIT Artifi cial Intelligence Laboratory, 1997.

Nagpal, Radhika. "Programmable Self-Assembly: Constructing Global Shape Using Biologically-Inspired Local Interactions and Origami Mathematics." PhD diss., MIT, 2001.

Nüsslein-Volhard, C. "Gradients That Organize Embryo Development." *Scientific American*, August 1990, 54–55, 58–61.

第 11 章　蒙蒂·代诺

Almási, George, Călin CaŞcaval, José G. Castaños, Monty Denneau, Derek Lieber, José E. Moreira, and Henry S. Warren Jr. "Dissecting Cyclops: A Detailed Analysis of a Multithreaded Architecture." *SIGARCH Computer Architecture News* 31, no. 1 (2003): 26–48.

Beetem, John, Monty Denneau, and Don Weingarten. "The GF11 Supercomputer." In *Proceedings of the 12th Annual International Symposium on Computer Architecture*, 108–15. Los Alamitos, CA: IEEE Computer Society Press, 1985.

第 12 章　大卫·肖

Bowers, Kevin J., Ron O. Dror, and David E. Shaw. "Overview of Neutral Territory Methods for the Parallel Evaluation of Pairwise Interactions." *Journal of Physics: Conference Series* 16 (2005): 300–304.

Shaw, David E., Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Ierardi, Istvan Kolossvary, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. "Anton, a Special-Purpose Machine for Molecular Dynamics Simulation." *Communications of the ACM* 51, no. 7 (2008): 91–97.

第 13 章　乔纳森·米尔斯

Karplus, W. *Analog Simulation.* New York: McGraw-Hill, 1958. Mills, J. "The Nature of the Extended Analog Computer." *Physica D* 237 (2008): 1235–56.

Mills, J. W., M. Parker, B. Himebaugh, C. Shue, B. Kopecky, and C. Weilemann. "'Empty Space' Computes: The Evolution of an Unconventional Supercomputer." In *Proceedings of the 3rd Conference on Computing Frontiers*, 115–26. New York: Association for Computing Machinery, 2006.

Rubel, L. A. "The Extended Analog Computer." *Advances in Applied Mathematics* 14 (1993): 39–50.

第 14 章　斯考特 · 阿伦森

Deutsch, David. *The Fabric of Reality: The Science of Parallel Universes and Its Implications*. New York: Penguin, 1998.

Grover, Lov. “A Fast Quantum Mechanical Algorithm for Database Search.” In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–19. New York: ACM, 1996.

Shor, Peter W. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.” *SIAM Journal on Computing* 26 (1997): 1484–1509.

关注图灵教育 关注图灵社区

iTuring.cn

在线出版 电子书《码农》杂志 图灵访谈……



QQ联系我们

读者QQ群：218139230



微博联系我们

官方账号：@图灵教育 @图灵社区 @图灵新知
市场合作：@图灵袁野
写作本版书：@图灵小花
翻译英文书：@李松峰 @朱巍ituring @楼伟珊
翻译日文书或文章：@图灵乐馨
翻译韩文书：@图灵陈曦
电子书合作：@hi_jeanne
图灵访谈/《码农》杂志：@李盼ituring
加入我们：@王子是好人



微信联系我们



图灵教育
turingbooks



图灵访谈
ituring_interview

如今，快速发展的不仅仅是计算机科学，金融、医药、航天及其他工程领域同样在飞速发展。航天工程师们正在设计能够适应极端天气及辐射的机器，一位科学家的“扩展模拟计算机”用测量电压的方式来取代传统计算，建立在DNA或细菌细胞基础上的“湿件”加工工艺也在竞相接近现实。这些别开生面的故事将读者带入探索未来智能机器的旅程，展现了计算机技术非常诱人的发展前景。

继成功打造《奇思妙想：15位计算机天才及其重大发现》之后，计算机科学家丹尼斯·萨莎与自由撰稿人凯茜·拉瑟再度联手，向我们揭示了计算技术不可思议的未来发展。作者深度采访了16位致力于解决计算领域前沿问题的顶尖科学家，介绍了他们的生平与成就，进而向读者展示一个异彩纷呈的计算世界。本书描述的计算和自然融合的技术，一定会让这个世界变得更美好。

“这部名人传记可谓‘袖珍’，每人篇幅不过寥寥数页。有关技术细节的东西都被单列了出来，除非读者情愿钻研，否则不会感到一点晦涩。”

——《华盛顿邮报》

“两位作者在这本书里讲解了分析蛋白质折叠必需的计算，这对于生物学研究特别是制药极为重要。为此，时间必须缩短到以飞秒计，1飞秒仅有1秒的一千万亿分之一。每一飞秒，反应涉及的原子交互都要计算清楚，然后是下一飞秒，如此持续。不难想见，模拟自然计算的每时每刻需要多么大的计算量和计算资源。”

——David Foster，芝加哥小子博客（Chicago Boyz）

“这是一本面向当前计算机设计和软件趋势的通俗读物。非专业人士一定会被它吸引，一步一步走近计算的最前沿。”

——《出版人周刊》

图灵社区：iTuring.cn
热线：(010)51095186转600

分类建议 计算机/IT人文

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-36311-4
定价：45.00元

看完了

如果您对本书内容有疑问，可发邮件至contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱：ebook@turingbook.com。

在这里可以找到我们：

微博 @图灵教育 : 好书、活动每日播报
微博 @图灵社区 : 电子书和好文章的消息
微博 @图灵新知 : 图灵教育的科普小组
微信 图灵访谈 : ituring_interview，讲述码农精彩人生
微信 图灵教育 : turingbooks